A Comparative Analysis of Reduction Algorithms in Picasso Tool for Cost Based Query Optimizers of Relational Databases

Neeraj Sharma[1] and Yogendra Kumar Jain[2]

[1] Dept. of CSE, Trinity Institute of Tech. and Research,
Kokta By-Pass , Raisen Road, Bhopal - 462 038 (M.P.), India
neeraj.bpl83@gmail.com
[2] Dept. of CSE, Samrat Ashok Technological Institute,
Civil Lines, Vidisha-464001 (M.P.), India
ykjain_p@yahoo.co.in

Abstract. Query optimization is a process of selecting an optimal Query Execution Plan from a number of plans available for execution of query which is very critical to the performance of a relational database. Picasso is a Query Optimizer analysis tool developed in the Database lab of Indian Institute of Science [24]. Using Picasso we can visualize the query execution plans and can implement a technique known as Plan Diagram Reduction [15][16][17] which can effectively increase the Query Optimizer performance. In this paper we briefly introduce the query optimization concept and then perform an exhaustive analysis of the reduction algorithms and try to establish some hard fact about their relative performance and reduction efficiency.

Keywords: Query Optimization, Selectivity, Plan Cardinality, Plan Diagrams, Checkpoints, TPCH.

1 Introduction

Structured Query Language (SQL) follows a declarative paradigm which means that the order of execution of instructions has to be determined by the SQL compiler. For this purpose the initial SQL query submitted by user is first converted into its equivalent relational algebra equation and for this expression a canonical query execution tree is generated. From this canonical tree many query execution plans or QEPs [3] can be generated using multiple techniques. Each query execution plan specifies a different order of execution of query with different set of operation. The task of Query optimizer is to search for the best possible query execution plans out of all these query execution plans. Query Optimizer plays a crucial role in determining the efficiency of Database systems. If the choice of query execution plans is not correct then the response time of the query processor will degrade. This increase in response time can be frustrating for the end user especially when the user is querying a large database in the likes of data warehouses or databases for decision support systems. All these requirements make Query Optimization a non trivial task for every commercial database management systems.

N. Meghanathan et al. (Eds.): CCSIT 2012, Part II, LNICST 85, pp. 176–185, 2012.

1.1 Picasso Database Query Optimizer Visualizer

Picasso [24] is a database query optimizer visualizer which generates cubical diagrams showcasing all the query execution plans that can be used for execution of a query in a specified selectivity space.

The diagram in Fig.1 (a) is a basic Picasso Diagram which shows 109 different plans that can be used for execution of query Q8 of TPCH Database. Each plan is represented using a different color. A plan is optimal in the area covered by its color in the plan diagram. The region of the plan diagram covered by a specific plan corresponds to the selectivities of the two base relations for which the plan will be optimal.

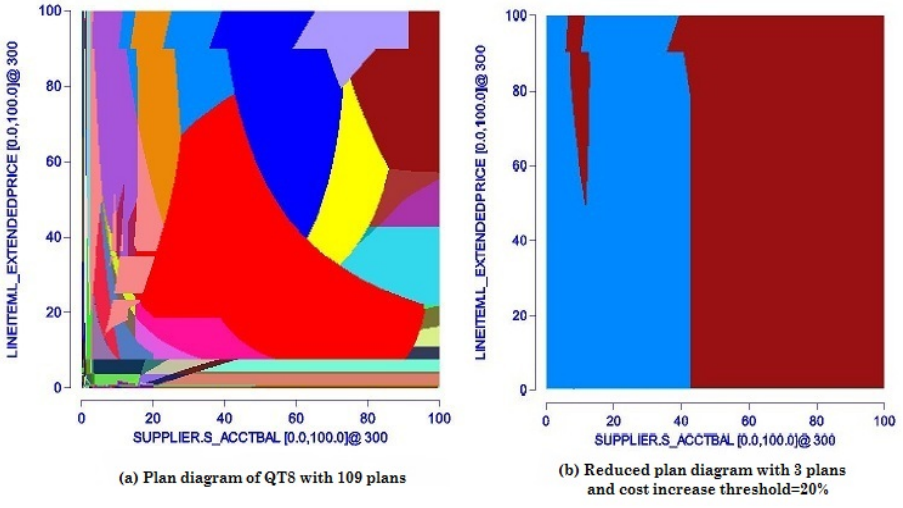

Fig. 1. Plan Diagram and Reduced Plan Diagram for QT8 of TPCH

In this paper we discuss one of the main diagrams generated by Picasso which is known as the Reduced Plan Diagram. Reduced plan diagram shows a reduced number of query execution plans that can be used for execution of query. This reduced plan diagram can be effectively used for increasing the performance of Query Optimizer. The plan diagram in Fig. 1(a) shows 109 different plans to execute QT8 which are reduced to 3 plans in Fig. 1(b). This reduction will help a lot in increasing the performance of query optimizer by decreasing the searching time for the optimal plan and decreasing the chances of selection of wrong plan.

2 Problems in Query Optimization

Query optimization is a hard problem [4]. The selection for best execution plan is done by using many different techniques of which the prominent ones are using

heuristics, some cost formulae, and randomized algorithms or genetic techniques [12]. Most Relational databases use heuristics combined with some cost formulas [22].

It is very time consuming to calculate cost of each and every execution plan to find the most optimal plan and is practically not feasible. Another problem faced by optimizers is the changing selectivity of base relations. The selection of optimal plans is based on the basis of some complex cost functions whose major parameters are the selectivities of base relations. It happens frequently that the estimated selectivities change. These wrong estimates of selectivities will lead to a poor choice of QEP.

2.1 Related Work

System R - A breakthrough work in query optimizers appeared in System R [1]. System R optimizer was the most rudimentary form of query optimizers. The cost function is based on parameters like disk access time and frequency, relation cardinality and number of tuples per page etc. The work on System R optimizer was further elaborated in [2] which discussed how the access path will be selected for execution of basic queries and for complex queries involving JOIN operations.

Eddies - The first ever technique for dynamic query optimization was discussed in [6]. The authors discuss a pure continuous adaptive query processing mechanism named Telegraph which checks for selectivities during "on the fly" time of query. Telegraph overlaps the optimization and execution phases. The initial optimal plan selected for executing a query is based on the cost functions whose parameters are determined from the system catalog. These parameters are subjected to change during run time of query. When such changes in parameters are discovered eddies try to reorder the operators used in the execution plan so as to minimize the execution cost.

Parametric Query Optimization (PQO) - System Rs algorithms were modified to generate multiple optimal plans for query execution and the process was called Parametric Query Optimization or PQO [7]. In PQO multiple candidate QEPs are generated for a query, each of which is optimal for some region of the parameter space. This collection of optimal candidate plans is known as Parametrically Optimal Set of Plans or POSP. Any one plan out of these POSP is used as final QEP during run time depending on the run time values of the parameters.

One significant problem with the PQO technique was that while dealing with piecewise linear functions, the solution proposed is pretty much intrusive. For this the authors of PQO suggests a modified optimization technique for Non Linear cost functions. This is known as AniPQO (Almost Non Intrusive PQO) [8].

Dynamic Optimization – Dynamic query optimization is a modern way for optimizing queries. In [4] a compile time dynamic plan optimization technique is described. Most dynamic optimization techniques use a dynamic programming model which compares cost of two plans and ignores the expensive ones. This requires a total ordering of plans. But authors of [4] suggest a "check plan" operator which maintains a partial ordering of plans.

The main problem of dynamic query optimization as pointed in [5] is the time spent in repeated collection of run time statistics like selectivities and resource availability and the searching of substitute plans. Algorithm described in [5] provides specific points in the query execution plan and the run time statistics are collected and substitute plans are searched only at these points thus minimizing optimization time.

The most practical approach in dynamic query optimization occurs in [9] in which a generalized and practically effective technique for optimization is suggested using CHECK operators. CHECK operators were placed at specific points in a query execution plan and they check the input cardinality. If the cardinality is within the specified range the current query execution plan is retained else the plan is changed during the run time.

CHECKS increase performance but excess of checks increase the execution time. So there is a risk and opportunity tradeoff which makes it important to determine how many CHECKS are to be inserted and where to be inserted. Five different rules for placement of CHECK operators in the Query Execution Plan were suggested by authors.

3 Plan Diagrams Reduction

The biggest motivation for reducing the number of plans arises after observing the skewed distribution of plans in the plan diagrams for different queries. The amount of skewed distribution of plans can be converted into a constant value known as Gini Index [25] which is a standard economic measure of income inequality.

When plan diagrams are generated with Picasso tool the Gini Index values are also generated and surprisingly the Gini Index value of most of the plans diagrams was greater than 0.75 substantiating the skewed distribution of plans over the selectivity space. These high values of Gini Index indicate that a big percentage of plans that cover very small space in the plan diagram need not be considered for execution of query. This is because these plans are highly vulnerable to poor execution times if the selectivities change even by a small order during the execution of query. Thus we can ignore or remove these plans to be considered as a possible candidate for execution of query.

A method used to substitute these plans with other plans which have high plan area coverage is known as *Plan Swallowing* [16][17]. The idea is that these smaller plans can be completely swallowed by their larger sibling plans which will effectively reduce the total number of plan cardinality in the plan diagram. There are two advantages of this approach. Firstly, it will reduce the searching time as the plan cardinality reduces and secondly, it will help us select robust plans which have higher plan space coverage and thus can tolerate higher variations in the plan selectivity. The process of plan swallowing may increase the cost of execution of a query but this increase in cost is controlled by user. This is represented as cost increase threshold (λ). Cost increase threshold (λ) of 10 is also sufficient but as proven in [14], a 20% cost increase (λ) can reduce the number of queries to an anorexic value which can significantly decrease the searching time of optimizers.

3.1 Algorithms Used for Reduction

Picasso uses three different algorithms for reduction. Initially COST GREEDY and AREA GREEDY were introduced in Picasso version 1 of which COST GREEDY proved to be more efficient. In Picasso V2 a new robust plan reduction algorithm SEER (Selectivity Estimate Error Resistance) [18] was introduced which performed plan reduction and gave robust plans which can withstand run time changes in the selectivities.

Cost Greedy - This algorithm operates under the assumption that the Cost Domination Principle holds and therefore only plan swallowing possibilities in the first quadrant are considered with respect to the plan under consideration. Cost Greedy ensure that the replacement plans are within the cost-increase-threshold at all points in the optimality regions of the replaced plans. The complexity of this algorithm is O(mn) where n is the number of plans in the diagram and m is the number of query points.

SEER - (Selectivity Estimate Error Reduction) - Due to wrong selectivity estimates the performance of the replacement plan could be much worse than the replaced plan. This problem naturally leads to the concept of a **robust** replacement – that is, a replacement where the λ-threshold criterion is satisfied at all points in the selectivity space, i.e. the replacement ensures **global safety**. For this we use two implementations of SEER:

Corner Cube-SEER - CC-SEER implements a more conservative test for robust plan replacement applying Abstract-plan-costing operations at the corner hyper-cubes of the selectivity space and is therefore significantly faster than the original SEER. Moreover, its performance is resolution independent unlike SEER, and therefore the performance gap between CC-SEER and SEER increases with higher resolution diagrams. Experimental results [16] indicate that CC-SEER's reduction quality is comparable to that of SEER instead of it being more conservative.

Lite SEER - Lite Seer is a light-weight heuristic-based variant of SEER that makes its replacement decisions solely based on Abstract-plan-costing operations at the corners of the hypercube, and is therefore extremely efficient. Lite SEER is optimal in the sense that it incurs the minimum work (complexity-wise) required by any reduction algorithm. While it does not guarantee global safety, experimental results [16] indicate that in practice, its safety and reduction characteristics are quite close to that of SEER and CC-SEER.

4 Experimental Analyses

Test Bed Environment

The TPC-H database was created using the "dbgen" software supplied with the TPC-H decision support benchmark [23]. A gigabyte-sized (1 GB) database was created on this schema and TPC-H queries templates were used.

An Intel Core i3 CPU M370 running at 2.4 GHz with 2 GB of main memory and 240 GB of hard disk, running 64 bit version of Windows 7 Home Basic operating system, was used in our experiments. The relational engine used is Microsoft SQL Server 2008.

4.1 Comparison of Reduction Algorithms

We performed experimental analysis of the three algorithms and compared their performance and plan reduction efficiency. Analysis is performed in two different parts. In the first part we compare the time taken by the three algorithms for performing the reduction. Then we compare the reduction efficiency of the three algorithms.

Computational Efficiency

Figure 2, 3, and 4 shows the time taken for reduction of plan diagrams by the three algorithms for 2D queries with plot resolution 10 and 3D queries with plot resolution of 10 and 30 respectively.

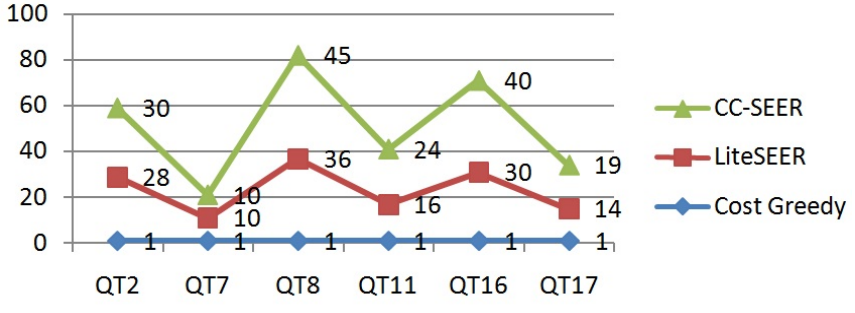

Fig. 2. Reduction time for 2D Queries, plot resolution=30, λ =10 (values in seconds)

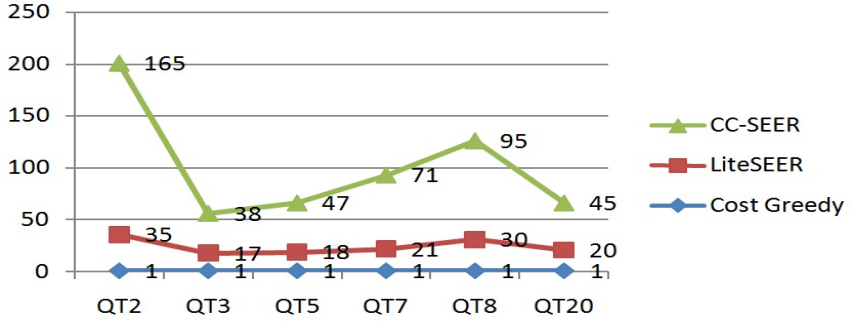

Fig. 3. Reduction time for 3D Queries, plot resolution=10, λ=10 (values in seconds)

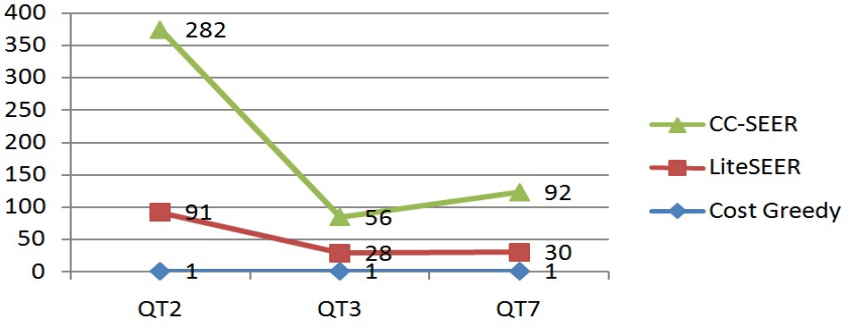

Fig. 4. Reduction time for 3D queries, plot resolution=30, λ=10 (values in seconds)

It is interesting to note that the reduction time taken by Cost Greedy algorithm was almost constant throughout the analysis for any combination of query and plot resolution. This makes Cost Greedy the fastest reduction algorithm. Next is LiteSEER which generates second best timings and then comes CC-SEER with the highest time readings. Conceptually comparison of SEER variants with Cost Greedy is not fair because SEER provides high quality and robust reductions. But when simple plan reductions are required, Cost Greedy proves to be much better than the two SEER algorithms.

Reduction Efficiency

Now we check the reduction efficiency by comparing the number of plans retained after performing reduction by the three algorithms. This analysis is also carried in three parts: Figure 5 shows the number of plans retained for 2D queries with plot resolution of 30 and figure 6 and 7 shows the number of plans retained for 3D queries with plot resolution of 30 and 10 respectively.

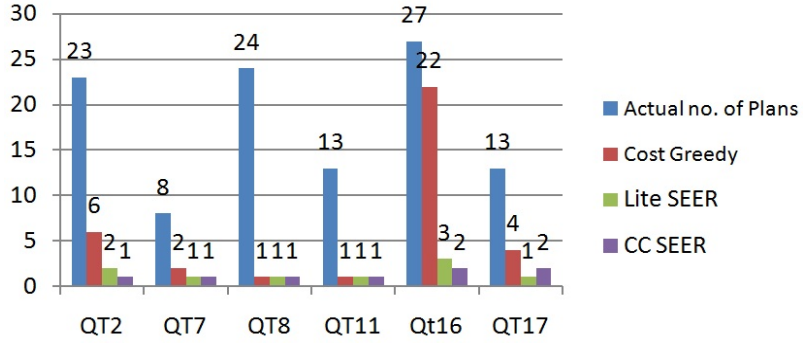

Fig. 5. Plans remaining after reduction of 2D Queries, Plot Resolution=30, λ=10

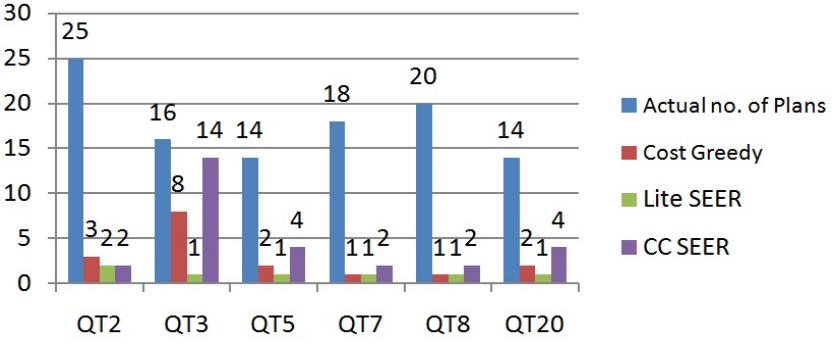

Fig. 6. Plans remaining after reduction of 3D Queries, plot resolution=10, λ=10

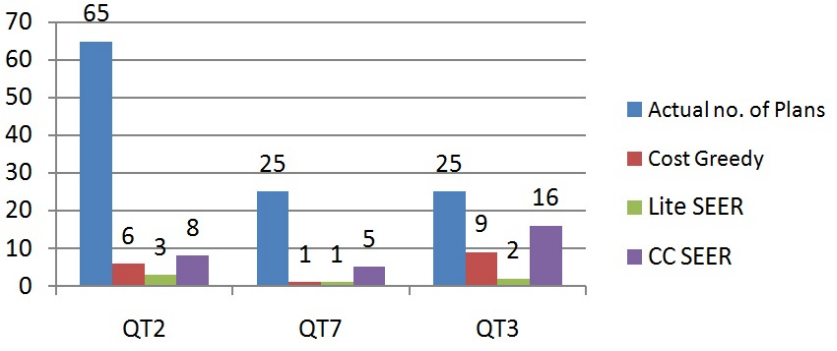

Fig. 7. Plans remaining after reduction of 3D queries, plot resolution=30, λ=10

Values shown in graph are the number of plans retained after reduction was carried. The number of plans before reduction was substantially high. The observation of these three tables and corresponding graphs gives a very interesting insight into the reduction efficiency of three algorithms. The reduction patterns for 2D queries were quite different from the reduction patterns of 3D queries. The following observations were made:

- In Reduction of 2D queries, the final plans retained were different for each algorithm in most of the cases. For example, reduction of QT8, QT11. The plans retained in LiteSEER and CC-SEER were missing from Cost Greedy which proves the quality difference between the three algorithms. Thus the normal assumption about the three algorithms holds true in case of 2D queries and CC-SEER gives robust plans which mostly are missing from the output of Cost Greedy and many times from LiteSEER as well.
- In case of 3D queries the observation was quite opposite. The plans retained in CC-SEER were also present in the list of plans retained in Cost Greedy and LiteSEER. The most extreme observation is that CC-SEER retained some plans which were initially not included in Cost Greedy and LiteSEERs list. This clearly indicates that CC-SEER is not a good choice for reduction of 3D queries because Cost Greedy and LiteSEER already produced the same diagrams in very less time.

5 Conclusion and Future Works

Query optimization is a difficult but a critical process for the database optimizers. Picasso introduces a novice way of plan diagram reduction which decreases the search complexity of query optimizers and also produces robust plans which improves the performance and reliability of query optimizers. Three such algorithms were discussed and the performances were compared. Results proved that Cost Greedy is the best algorithm in terms of execution time but if reliability was desired then Lite SEER and CC-SEER were better. For 2D queries CC-SEER must be used in spite of its long execution time but for 3D queries LiteSEER and Cost Greedy proved better than CC-SEER.

There are many interesting future works to be carried. Cost Greedy and SEER based algorithms are purely compile-time approach and it can be used in conjunction with run-time techniques such as adaptive query processing [13] for addressing selectivity errors in the higher nodes of the plan tree. Another improvisation in the design of these algorithms can be to include the technique of CHECKS suggested in [9] which can further increase the quality of plans produced after reduction. Lastly it would be interesting to use these algorithms on the upcoming TPCE dataset and some other dataset having queries with more than 3 dimensions.

References

1. Astrahan, M.M., et al.: System R: Relational Approach to Database Management. ACM Transactions on Database Systems 1(2) (1976)
2. Selinger, P., Astrahan, M., Chamberlin, D., Lorie, R., Price, T.: Access Path Selection in a Relational Database Management System. In: SIGMOD (1979)
3. Freytag, J.C.: Basic principles of query optimization in relational database management systems. In: Proceedings of IFIP Congress (1989)
4. Cole, L., Graefe, G.: Optimization of dynamic query evaluation plans. In: SIGMOD (1994)
5. Kabra, N., DeWitt, D.: Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. In: SIGMOD (1998)
6. Avnur, R., Hellerstein, J.M.: Eddies: Continuously Adaptive Query Optimization. In: SIGMOD (2000)
7. Hulgeri, A., Sudarshan, S.: Parametric Query Optimization for Linear and Piecewise Linear Cost Functions. In: VLDB (2002)
8. Hulgeri, A., Sudarshan, S.: AniPQO: Almost Non-intrusive Parametric Query Optimization, for Nonlinear Cost Functions. In: VLDB (2003)
9. Markl, V., Raman, V., Simmen, D., Loman, G., Pirahesh, H., Cilimdzic, M.: Robust Query Processing through Progressive Optimization. In: SIGMOD (2004)
10. Reddy, N., Haritsa, J.: Analyzing plan diagrams of Database query optimizers. In: VLDB (2005)
11. Aslam, M.: Picasso: Design and implementation of a Query Optimizer Analyzer. Master's Thesis, Dept. of Computer Sci. and Automation, IISc (2006)
12. Kader, R.A., van Keulen, M.: Overview of query optimization in XML Database Systems. University of Twente Publications (2007)

13. Deshpande, A., Ives, Z., Raman, V.: Adaptive Query Processing. In: Foundations and Trends in Databases. Now Publishers (2007)
14. Harish, D., Darera, P., Haritsa, J.: On the Production of Anorexic Plan Diagrams. In: VLDB (2007)
15. Darera, P.: Reduction of query optimizer Plan Diagrams. Master's Thesis, Supercomputer Education & Research Centre, IISc (2007)
16. Harish, D., Darera, P., Haritsa, J.: Robust plans through plan diagram reduction. In: VLDB (2007)
17. Harish, D., Darera, P., Haritsa, J.: Identifying Robust Plans through Plan Diagram Reduction. In: VLDB (2008)
18. Harish, D.: SIGHT and SEER: Efficient Production and Reduction of Query Optimizer Plan Diagrams. Master's Thesis, Dept. of Comp. Sci. and Automation, IISc (July 2008)
19. Dey, A., Bhaumik, S., Harish, D., Haritsa, J.: Efficiently Approximating Query Optimizer Plan Diagrams. In: VLDB (2008)
20. Haritsa, J.: The Picasso Database Query Optimizer Visualizer. In: VLDB (2010)
21. Haritsa, J.: Query optimizer plan diagrams: Production, Reduction and Applications. In: ICDE (2011)
22. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems, 5th edn. Addison-Wesley (2008)
23. Transaction Processing Council, `http://www.tpc.org/tpch`
24. Project Picasso, IISC, `http://dsl.serc.iisc.ernet.in/projects/PICASSO/index.html`
25. `http://en.wikipedia.org/wiki/Gini_coefficient`