EasyChair Preprint

№ 8331

Advantages of Anytime Algorithm for Multi-Objective Query Optimization

Rituraj Rituraj and Annamária Várkonyi Kóczy

June 21, 2022

Advantages of Anytime Algorithm for Multi-Objective Query Optimization

1st Rituraj Rituraj
Doctoral school of applied informatics and applied mathematics
Obuda University
Budapest, Hungary
rituraj88@stud.uni-obuda.hu

2rd Annamária R. Várkonyi Kóczy
Institute of automation, Kandó Kálmán Faculty of Electrical Engineering
Obuda University
Budapest, Hungary
varkonyi-koczy@uni-obuda.hu

***Abstract*— Data is becoming an important source of information these days. According to the recent Forbes survey, there are 2.5 quintillion bytes of data created each day at our current pace. While getting these amounts of data, it is necessary to process them in order to extract information. These data can be accessed in many ways in order to get meaningful information. This paper only deals with the query plans model through multi-objective optimization process using anytime algorithm. Query plans is an ordered stairway used for accessing data in SQL relational database systems. Query plans provides diverse tradeoff between conflicting cost matrices. The cost matrices are execution time, energy consumption and execution fees in multi-objective aspects. When SQL database run the queries by choosing an optimum query execution plan then it minimizes the query cost, which is very crucial for the query optimizer. Multi-objective query optimization and anytime algorithm possess very specific properties in order to support an interactive process which dynamically add various constraints and then finally select the best plan based on the continuously refined visualization of optimal cost tradeoffs. First, the anytime algorithms generate the multiple result plan sets which increase the quality with low latency rate between consecutive results. Second, the consecutive results will be incremented to avoid regenerating query plans when being invoked several times for the same query but with different user constraints. This paper deals with the advantages of anytime algorithm for the multi objective query optimization to analyze the complexity which offers an attractive tradeoff between the results. It can be used to update frequency, single invocation time complexity and multiple time over invocations. These properties make anytime algorithm suitable to be used within an interactive query optimization process.**

Keywords*— Query plans, Multi-objective optimization, Anytime algorithm.

I. INTRODUCTION

Traditionally, we cape of long cost matrices in query processing execution time. However, nowadays we have many scenarios which need to know about multiple cost matrices to trade between them. This uses the cloud computing. Cloud computing in terms of tradeoff execution time is against monitoring fees. If we rent more, then all the providers may refuse the execution time and set for optimal cost tray of the given query [1]. This is also called ParetoFrontier. In order to trade precisely against the execution time, there are three cost matrices i.e. execution time, execution fees, and result precision [2]. There are few examples like:

- Concurrent systems: - System resources (cores, buffer spaces), execution-time
- Energy- Aware computing: - Energy, execution-time
- Crowd sourcing: - Fees, execution-time, accuracy.

In summary, many scenarios of query processing are a tradeoff between multiple cost matrices. It has a fundamental effect on the query optimization problems. This is because traditionally we have only one cost matric and the goal is to minimize it. Nowadays, we have many matrices and the goal is to find the best tradeoff out of them to integrate results of user preferences into the optimization process. It is because some users look for execution time, but some look for execution fee [3].

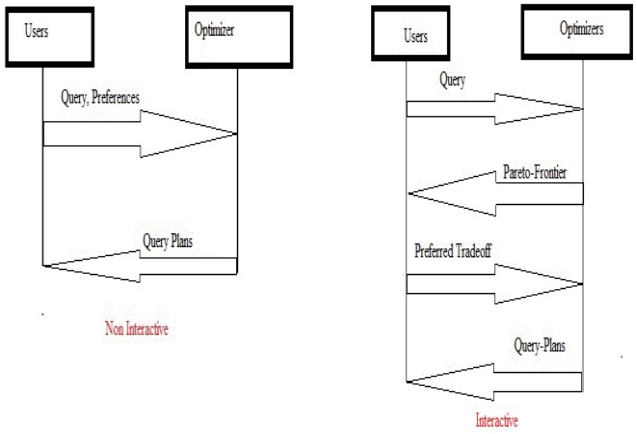

Fig 1. Relation between Non-interactive and Interactive Approaches

In non- interactive approach, the user basically specifies the preferences to get of the query as the problem input to the optimizer. After this the optimizer will analyze those preferences in order to find the optimal query output. The problem is that the queries are generally very difficult to formalize these preferences. The users don't know what they want before seeing it. So, from the users prospective and for much more convenience, interactive model is approached [4]. At the first stage of this model, the user uses query to the optimizer where optimizer visualizes result of optimal cost matrices. After this it introduces a preferred tradeoff out of it. The problem here is efficiency. It needs to calculate the complete pareto-frontier which is sometimes not feasible. It might take an hour for simple query. It can approximate the real pareto-frontier, but it might take a minute and form an interactive interface. This is still a long process. For that reason, in the second stage user thought to increment the life on the optimization process. This means that it divides the optimization process into many small incremental steps. After each step it provides an intermediate optimization results to the user. This gives the possibilities to dynamically specify cost-bounds for the optimizer. This helps in order to guide the optimization towards positive result space. This is an anytime algorithm because it doesn't return one approximation of the pareto-frontier but multiple approximation of increasing qualities. There are incremental algorithms since it takes of avoiding reductant graph over multiple approximation. It always approximate pareto-frontier for the same query multiple times. If it doesn't pay attention, then it might have many faults that will regenerates multiple times for same instance [5]. In the figure 2, the user has issues to query plans and the optimizer very quickly generates a cross -strait approximation of the pareto-frontier. If the user doesn't do anything then these approximations is refined, and user wants

to restrict the execution fee. In this case, it can dynamically identify the cost-round and the optimization will end. After this, it will only be focusing on the area near to the cost bound. This area is obtained by approximating the pareto frontiers by using the anytime algorithm method. Finally, if the user feels that the approximation is sufficiently closed to the desired output then the user clicks on those pareto-frontier to get the execution time and fee [6].

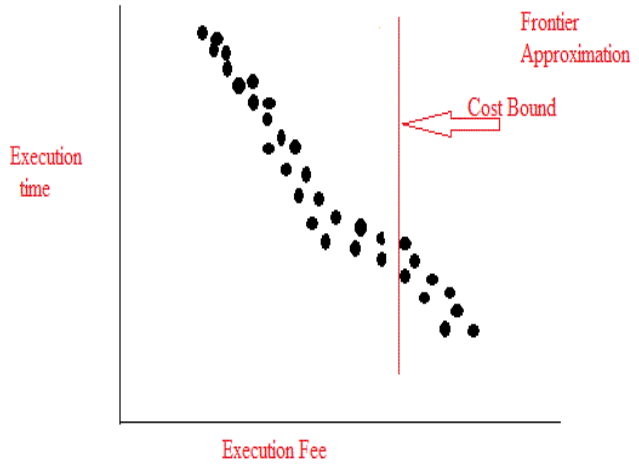

Fig 2. Anytime Algorithm Optimization Example

The scenarios for multi-objective query optimization in ascending order of time constraints are optimization before run time, optimization at run time, and optimization at run time and interactive

II. ADVANTAGES OF ANYTIME ALGORITHM FOR OPTIMIZER

Anytime algorithm is the algorithm which handle the case of too many abrupt changes and their consequences in the signal processing, monitoring, diagnostics, or larger scale embedded systems [7]. The algorithm finds better and better solutions as it keeps running for longer duration. In query optimization process, the interactive optimizer plays an important role in approaching the anytime algorithm. The user interface takes the action of the user and control the resolution and refinement. The incremental optimizer specifies that the pareto-frontier should be approximated for the given resolution and cost bound. Now the optimizer is incremented because it makes a set of query plans across convocation and generate result plan. The generated result plan might be considered later when the resolution gets refined and cost bound changed. The goal of optimizer design is to avoid redundant work and keep optimization time proportional to current resolution and bound. On high level, steps per optimizer invocation includes a) retrieve candidates and prune, in the first phase b) generate plans and prune, in the second phase. Query are the syntactically valid parse trees whose semantic meanings are reasonable and need to interpret [8].

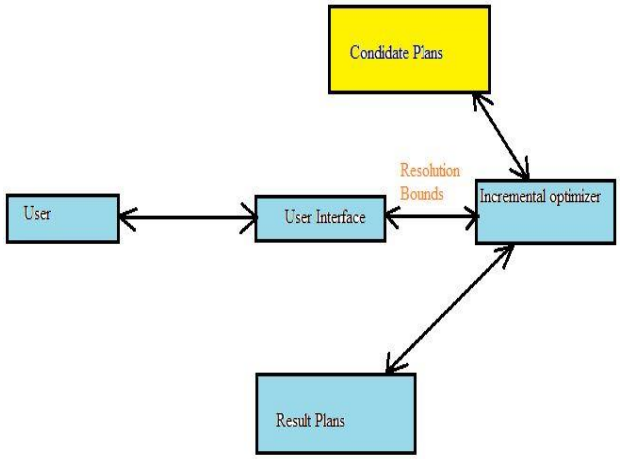

Fig 3. Overview of Interactive Optimizer

The main problem which arises on the high level is the generation of plans twice. In order to avoid it, we need to be careful in combining the plans. If we combine the one new plan to another new plans, then the plan will generate twice. Therefore, we must join the old plan with the new plan. In such cases the optimize query plans do not contains any similar items. This algorithm used to compute policies for decision problems represented as multi- stage influences. It helps to constructs policies incrementality which helps to construct policies with more information available to the decision make at each step [9].

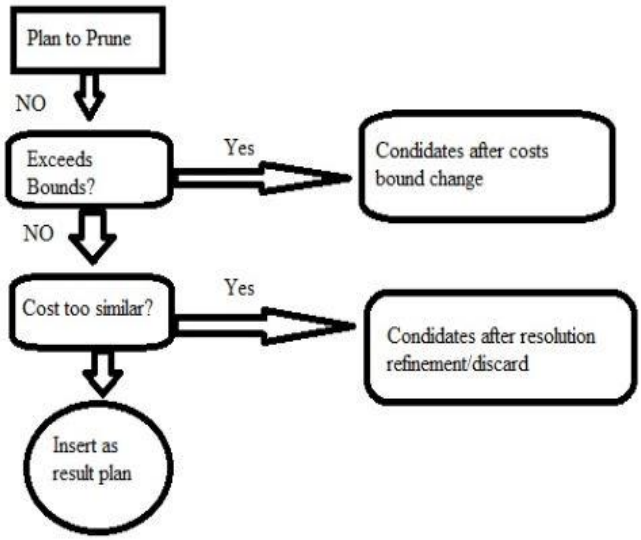

Fig 4. Showing the Pruning Plans

Parallel correctness serves as a framework for studying correctness. The implications of data partitioning are one-round query process in evaluation algorithms. The algorithm is useful in describing the compilation and monitoring mechanism for the intelligent systems. It can control deliberation time of the system. The algorithm is ideal for bounded time pathfinding problems. It helps in finding the feasible sub-optimal solution very quickly and improving it until time runs out. Anytime rectangle expansion algorithm used to run an accelerated sub-optimal search and then repairs the sub-optimality. It provides narrower and more accurate sub-optimality bounds of its solutions [10].

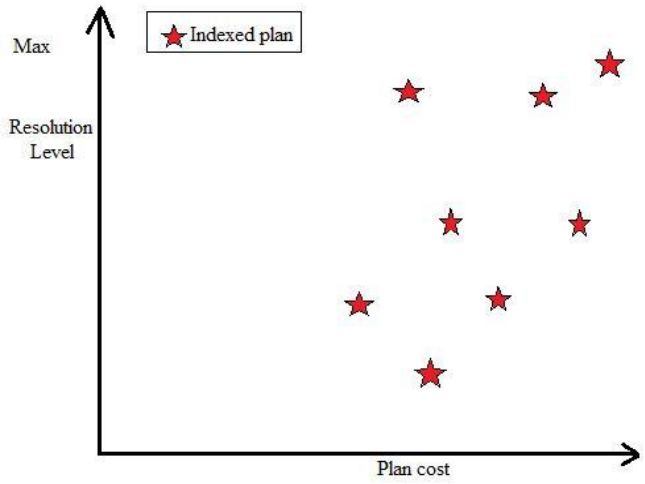

Fig 5. Showing Retrieving Plans

Anytime algorithm improves query plans via a multi-objective version of hill climbing. This applies multiple transformations in each climbing step for maximal efficiency. In each iteration process of query optimization, anytime algorithm performs in expected polynomial time. This is based on an analysis of the expected path length between a random plan and local optima reached by hill climbing. This algorithm can optimize queries with hundreds of tables and performs other randomized algorithms with multiple cost matrices.

III. COMPLEXITY ANALYSIS

The analysis of the complexity of algorithm is a mutable process from the prospective of some similar query plans. After it generated, it is indexed as candidates and might be re-indexed as candidates couples of times before it is finally indulging as a result plan or discarded. Then it shows the whole life cycle happens only once over a series of optimization convocation [11]. The number of times depends on the re-indexed candidate which is bounded. Altogether it means that the amount of work that has on the query plan is bounded. The complexity analysis results in incrementing the optimization process of query plans. The impact of incrementality is a) the optimization time becomes incremental part and moderately overhead b) optimization time proportional to search space size [12].

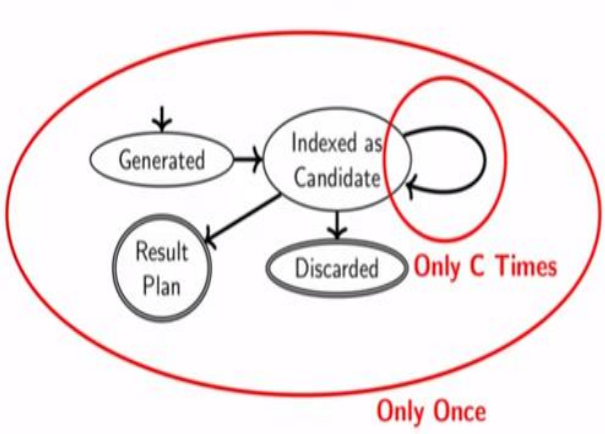

Fig 6. Query Plan Life Cycle

Anytime algorithm in the query optimization process is used for the simultaneous coalition of the structured query plans and indexed candidates. This optimization problems have many real-world applications which include minimizing the total cost associated with the execution of a request. These associated costs are function of the access time cost for input and output of the system which is involved in accessing the physical data stored on disk. To evaluate the algorithm's performance, we extend established methods we extend established methods for synthetic problem set generation. The benchmark of the algorithm is using randomized data sets of varying distribution and complexity. The algorithm solves the problem of assigning query plans to regions in a major commercial strategy game. This shows that the algorithm can be utilized in query optimization to coordinate smaller sets of agents in real-time. Anytime algorithm computes a polytopic underapproximation of the stochastic result plans set. It synthesizes an open-loop controller using convex optimization [13]. Query optimization typically tries to approximate the optimum by comparing several common-sense alternatives. The complexity of the model can be tuned both by evaluating only a query model and by improving the existing model increasing the granulation in the knowledge of new information. Fuzzy logic with Tsukamoto inference system can be used in order to have a much faster query response time. This can accelerate query response time. Thus, by combining fuzzy and anytime techniques it is a possible way to overcome the difficulties caused by the high and explosive complexity of the applied models and algorithms [14].

IV. CONCLUSION

It is very difficult to formalize the user preferences; therefore, multi-objective query optimization should be an interactive process. Anytime algorithm is very helpful in designing the interactive optimizer for the whole process to be carried out within the limited time. The query optimization motivates incremental anytime algorithm. Fuzzy and anytime algorithm could be the better option for query optimization.

ACKNOWLEDGMENT

I heartily thank my Ph.D. supervisor Annamária R. Várkonyi Kóczy from the Obuda University for providing me the sufficient data and prior scientific papers to start with the topic. She continuously discussed the methods of writing a conference paper in the first semester of Ph.D. I also thank my friends Awaish Qadir, Neerendra Kumar and Yatish Bhatla for making me understand the topic and templates for writing the paper.

REFERENCES

[1] Trummer I, Koch C. An incremental anytime algorithm for multi-objective query optimization. InProceedings of the 2015 ACM SIGMOD International Conference on Management of Data 2015 May 27 (pp. 1941-1953). ACM.

[2] Stulova N, Morales JF, Hermenegildo MV. Some trade-offs in reducing the overhead of assertion run-time checks via static analysis. Science of Computer Programming. 2018 Apr 1;155:3-26.

[3] Blelloch GE, Gu Y, Shun J, Sun Y. Parallel write-efficient algorithms and data structures for computational geometry. InProceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures 2018 Jul 11 (pp. 235-246). ACM.

[4] Tian R, Qiu J, Zhao Z, Liu X, Ren B. Transforming query sequences for high-throughput B+ tree processing on many-core processors. InProceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization 2019 Feb 16 (pp. 96-108). IEEE Press.

[5] Kolchinsky I, Schuster A. Join query optimization techniques for complex event processing applications. Proceedings of the VLDB Endowment. 2018 Jul 1;11(11):1332-45.

[6] Leis V, Radke B, Gubichev A, Mirchev A, Boncz P, Kemper A, Neumann T. Query optimization through the looking glass, and what we found running the join order benchmark. The VLDB Journal—The International Journal on Very Large Data Bases. 2018 Oct 1;27(5):643-68.

[7] Goyal D, Pabla BS. Condition based maintenance of machine tools—A review. CIRP Journal of Manufacturing Science and Technology. 2015 Aug 1;10:24-35.

[8] Kossmann D, Ailamaki A, Balazinska M. SIGMOD Officers, Committees, and Awardees. SIGMOD Record. 2016 Mar;45(1):1.

[9] Horsch MC, Poole D. An anytime algorithm for decision making under uncertainty.). Morgan Kaufmann Publishers Inc. InProceedings of the Fourteenth conference on Uncertainty in artificial intelligence 1998 Jul 24 (pp. 246-255

[10] Li C, Bi W. ANYTIME RECTANGLE EXPANSION A* ALGORITHM FOR TIME-LIMITED PATHFINDING An Zhang. International Journal of Robotics and Automation. 2019;34(3).

[11] Prékopa A, Szántai T, Zsuffa I. Secondary Stochastic Processes and Storage Reservoir Optimization. Acta Polytechnica Hungarica. 2018 Jan 1;15(1).

[12] Dulai T, Dósa G, Werner-Stark Á. Multi-Project Optimization with Multi-Functional Resources by a Genetic Scheduling Algorithm. Acta Polytechnica Hungarica. 2018 Jan 1;15(4):101-19.

[13] Vinod AP, Oishi MM. Stochastic reachability of a target tube: Theory and computation. arXiv preprint arXiv:1810.05217. 2018 Oct 11.

[14] Várkonyi-Kóczy AR. Fuzzy approaches in anytime systems. InOn Fuzziness 2013, Springer, Berlin, Heidelberg (pp. 725-735).