Vector Evaluated Genetic Algorithm-Based Distributed Query Plan Generation in Distributed Database

Vikash Mishra and Vikram Singh

Abstract Distributed query processing (DQP) determines an optimal query plan, which generates user query results in efficient manner by selecting optimal set of database sites. Multi-objective DQP problems become more complex because a query optimizer has to select optimal, non-dominated QEP's, query equivalent plans, based on conflicting objective values. In past few years, evolutionary techniques are employed on such problems, although they are unable to get a good balance between efficacy and efficiency in all attempts. A meta-heuristic-based algorithm is presented which determines the combinations of database sites, in response to a query or group of queries. In this paper a technique is proposed for the optimal query plan generation, based on the meta-heuristics, modelled for distributed query processing, through an improved vector evaluated genetic algorithm for generation and selection of optimal query plans on distributed database. The algorithm's optimization performance is evaluated with other approaches and optimization reliability along with efficiency is benchmarked using performance graphs; comparisons indicate that the vector evaluated genetic algorithm (VEGA) converges better than aggregation-based method (weighted-sum approach). Top-K query plans, average query cost and number of generations are the parameters used for the comparative analysis.

Keywords Aggregation-based genetic algorithm · Bi-objective genetic algorithm · Distributed database · Distributed query processing · Top-K · Query vector evaluated genetic algorithm

Vikash Mishra (✉) · Vikram Singh
National Institute of Technology, Kurukshetra, Haryana, India
e-mail: mishravikash03@gmail.com

Vikram Singh
e-mail: viks@nitkkr.ac.in

N. Afzalpulkar et al. (eds.), *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing*,
DOI 10.1007/978-81-322-2638-3_37

Introduction

In modern history, the Internet in its all probability is the most desired and used distributed processing and computing environment and on the other hand, a user in this environment with limited capabilities for retrieving the desired result are at best. In such distributed processing environment, queries posed by user or applications routinely require gathering data from multiple sites or data sources [1]. Advanced database systems are embedded with the capability of various transparency mechanisms due to which as user never face or feel the effect of distribution of data and in such systems the data pertinent to a query move to a single server for final compilation. In traditional way, query on distributed processing environment is optimized centrally and executed synchronously [2]. Improving the performance of a database system is one of the key research issues from over the decades. Distributed database system (DDBS) is effectual means to improve reliability and availability of data to improve the overall performance of a processing system [3]. The users at local sites can work independently as well as communicate with other sites to retrieve data for answering global queries. Such a setup is referred to as a distributed database system (DDS) [4]. Query posed on a DDS is generally decomposed into secondary queries, which are executed at respective local sites with local data store, before communicated to central control sites for final result generation and finally at the user end, an integrated result is displayed.

A distributed query processing (DQP) strategy aims to minimize the overall cost of query processing in such systems [5, 6]. The cost of query processing in a DDS comprises two costs: the local processing cost and the site-to-site communication or the transmission cost of relation fragments. The total cost incurred in processing a distributed query can thus be taken as the sum of the local processing cost, cost of local operations on local data and cost of transferring locally processed data from participating sites to a central control sites. DQP iteratively evaluates all QEPs of a user query and determines the most optimal query plan that minimizes the total cost that is local processing (CPU, I/O) cost and communication cost [7]. In DDBS, the number of query equivalent plans (QEPs) grows at least exponentially through the increase in amount of relations accessed by the query, as relations are either replicated fully or replicated with partitioned sub-relations. Performing an exhaustive search for optimal QEPs over the all possible combinations of query plans is not viable due to a huge solution space. Therefore, in large DDB, devising a query processing strategy that optimizes the total query processing cost is shown to be a combinatorial optimization problem with NP-complete in complexity nature [8].

Over the last two decades, evolutionary algorithms have gained immense popularity due to their applicability in solving engineering optimization problems and complex scientific problems. These algorithms are inspired by the Darwinian evolution that accentuates the concept of “Survival of the Fittest”. It is, thus, metaphorical to the natural social behaviour and biological evolution of species. The evolutionary strategies are now proved to be the most proficient method of choice for solving such problems. Genetic algorithm-based techniques which belong to the

class of evolutionary algorithms have also been widely used in solving complex real-life science and engineering problems. The strength of GA as a meta-heuristic comes from its ability to combine the good features from several solutions to create new and better solutions over generations. Most real-world scientific and engineering problems have often conflicting and competing objectives that need to be optimized. In subsequent years, several different evolutionary algorithms (VEGA [9], NPGA [10], NSGA [11], NSGA-II [4], SPEA [11], SPEA-II [11], PAES [11], PESA [12, 11]) have effectively employed to solve the classic optimization problems.

This paper is organized as follows. Next section describes DQP and its motivation and proposed heuristic. Fundamentals of MOEA are discussed in subsequent section, also its suitability for DQP problem. Vector evaluated genetic algorithm (VEGA) is described in next section and an example illustrate the use of VEGA-based DQPG algorithm for generating optimal query plans. In graph section performance over other techniques in same section, in which comparison. In the last section conclusive discussion is presented with summary of approach.

Distributed Query Processing

In today's scenario, with a multifold increase in the size operational data leads to growth of size of DDBS, the communication cost asserts a major impact on the overall cost of query processing. The cost incurred in communicating data through a congested network path or the communication of large data units between sites with higher communication costs can highly influence the cost of query processing. It thus also plays a key role in determining the overall performance of a DDBS [4, 6]. There can be a number of possible ways to process and communicate sub-relations relevant to the user query. In difference to the centralized query processing, distributed query processing cannot statically generate result at one single site. In DQP the user query is executed at node and its neighbours [6, 13, 14]. The optimal query plan generation and selection greatly depend on the cost function, as shown in Fig. 1.

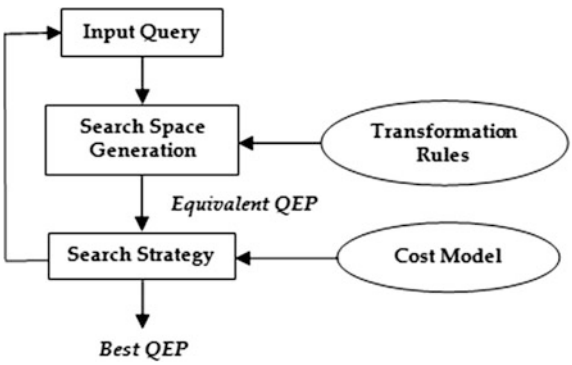

Fig. 1 Distributed query optimization

In distributed relational database, logical data is replicated either fully or partitioned way to achieve higher degree of reliability in environment and serve higher availability data for efficient processing closer to the user sites or locations. This leads query processing to complex optimizations scenario, as a number of QEPs are exponentially increased. An approach is proposed in this paper using genetic algorithm (GA) to generate the query plans keeping optimality among solutions at highest priory. Primarily, a query optimization focuses on to diminish cost or amount of data a query requiring from multiple logical sites for processing queries. As a result, computing optimal distributed query plans becomes a multifaceted problem. In [14] a distributed query plan generation (DQPG) approach for single-objective, query proximity cost (QPC) is proposed, where optimization among QEPs is according to the heterogeneity within a QEP on the order of sites used. We proposed a solution for DQPG problem, and analysed optimization performance by including additional design objective. In the following objectives are briefly discussed.

Motivation: DQP is a model to determine the optimal policy for processing any given user query. The model is based on operating cost (cost of local processing and communication), which is a function of processing sites for query operators and sequence of these database operators. In DQP quality of QEP is evaluated according to the processing time required by the QEP [15]. In DQP, cost of data transmission between sites is the dominating cost. The objective of distributed query processing is to minimize the data transmission among sites thereby reducing the data transmission cost. Semi-joins have been used [16, 17] to reduce the communication cost. Semi-join reduces the communication cost and exploits the parallelism for query plan execution. Semi-joins carry out this by reducing the amount of attributes or tuples/attributes transfers among sites for generating results, thereby reducing the communication cost. DQP is phenomenon driven by the heuristics [18].

Heuristic-1: Query proximity cost (QPC) quantifies the heterogeneity in a QEP; a QEP with least number of distinct sites is better as it leads to less data transfer among sites. In this case, more than one QEPs with higher number of same relation is considered better [13, 14]. The fitness function, i.e. the QPC, based on these two heuristics is given below:

$$\mathrm{QPC} = \sum_{i=1}^{M} \frac{K_i}{N}\left(1 - \frac{K_i}{N}\right)$$

where M indicates the total number of sites required in a QEP, K_i refer times the ith site in the QEP and finally N is the total number of relations required in QEP. GA uses the above fitness function to select the query plans having minimum QPC for the next generation. The selected query plans undergo genetic operator (crossover, mutation) for the generation of new pool.

Heuristic-2: Communication cost (CC), first of all algorithm forms the all possible trees assuming sites as nodes of tree, then evaluate minimum cost tree

among all. The communication path between two node is defined on the basis of data stored into the sites, such as the communication cost between node A and B is different in the case when we want move from B to A or A to B. The data from sites is transferred such a way that caused minimum cost to overall plan. So we are trying to minimize this heuristic. For the generation of the communication cost for query plan set, we need a communication cost table consisting communication cost of all site-to-site path.

Multi-objective Evolutionary Algorithm (MOEA)

These algorithms are inspired by the Darwinian evolution that accentuates the concept of "Survival of the Fittest" [19, 20]. It is, thus, metaphorical to the natural social behaviour and biological evolution of species. The evolutionary techniques are now proved to be the most proficient method of choice for solving such problems. Genetic algorithm-based techniques which belong to the class of evolutionary algorithms have also been widely used in solving complex real-life science and engineering and scientific problems [21]. The strength of GA as a meta-heuristic comes from its ability to combine the good features from several solutions to create new and better solutions over generations. Most real-world scientific and engineering problems have often conflicting and competing objectives that need to be optimized [22]. The evolutionary tactics are established as a best possible tool for many engineering and science related problems to optimize the different objectives and find efficient trade-offs unlike the classic techniques. The strength of GA as a meta-heuristic comes from its ability to combine the good features from several solutions to create new and better solutions over generations. Design objectives in most engineering and science scenario are disparate and contradictory for which optimized solution is required [20, 22]. A technique based on evolution is best fit for these kinds of problem, each evolution or iteration generated new improved solution and continues until certain degree of optimality is achieved over set of solutions. Evolutionary techniques are being used and suited for such problems on which concurrent optimization can be achieved over multiple objectives. The best part of GA emerges for multi-heuristic problems on its ability to combine the good features from several solutions to create new and better solutions over generations [20]. Most real-world scientific and engineering problems have often conflicting and competing objectives that need to be optimized. The evolutionary strategies are proved to be best suited for this class of problems as they can simultaneously optimize the different objectives and find efficient trade-offs unlike the classic techniques.

Many engineering and scientific problems cannot realistically modelled using single design objective, and thus multiple and often conflicting objectives are designed. Single genetic algorithm (SGA) is potentially unable to solve optimization problems with multiple conflicting objective functions, and this leads to a set of evolution of customized genetic algorithms; these are demonstrated and observed

effective on determining the excellent solutions. The solution set consists of a pareto front of optimal solutions for problem, and a solution in pareto front is considered best with respect to other solution for each of the objectives.

Schaffer (1985) considers the capability of single-objective GA (SGA) for multi-objective-based optimization problems and adapted a new variant of GA, named vector evaluated genetic algorithm (VEGA). In this new development, basic methodology of GA is same except the selection process of chromosomes, as selection is based performed on entire population by applying fitness values from each of the objectives, simultaneously and this generates set of population equal to the number of objectives considered. A problem having m design objectives and total population size of N, m subpopulations of N/m size is created. The selection step was modified to perform proportional selection at each generation for each of the objectives [15], and next for algorithm to proceed with the application of crossover and mutation in the standard mode.

VEGA is implemented over bi-objective query optimization, and performance of aggregation-based genetic algorithm (weighted-sum approach) with VEGA is analysed. The outcome of VEGA-based optimizations is better, as a aggregation-based GA has main demerits in complexity to decide the weights that can be appropriate to scale the objective, to accomplish this prior information which is required about problem, particularly if we consider that optimal point obtained will be purpose of these weights [23].

Vector Evaluated Genetic Algorithm (VEGA)

VEGA is first practical algorithm, implemented for the population with multiple properties or objectives. In [24] Schaffer adapted fundamental GA for multi-objectives by modifying the selection genetic operator of GA. In VEGA vector term represents the objective value for the multi-objective problem. This variant of genetic algorithm is a simplest and an uncomplicated adaption of single-objective genetic algorithm (SGA) for multi-objective optimization with modified selection genetic operator [25]. This algorithm divides total population into set of subpopulations equal to number of design objectives. Each subpopulation is created based on the fitness values of specific objectives, and for each generation, fitness assignment for first subpopulation is done according to the values of first objective function, and so on [24]. Selection techniques are applied on each of the subpopulations and based on the objectives, fitness value due to mating pool is created. In order to minimize positional partiality among the solutions by randomly shuffling before they are partitioned according to the objectives into set of subpopulations, this is useful in managing problems on which objective takes values of different domains and orders of magnitudes. As all QEPs of each of subpopulations is allotted fitness and restricted selection within each of subpopulations, this emphasizes better solution equivalent to respective objective function. Moreover, since neither two QEPs are compared on different objective functions

nor generate any complexity. In VEGA, any fundamental operator of selection can be used, and we used for a proportionate (Roulette wheel method) as selection operator. Selected QEPs are inserted into the mating pool for the further reproduction using genetic operators, like crossover using crossover probability (P_c) and mutation using mutation probability (P_m). VEGA iteratively evolves solutions until as stopping criteria met or fulfilled these criteria values are provided by decision maker or user [26].

(a) *Algorithm*

Input: Query plans {query$_1$ [], query $_2$[]…}
Output: New set of Query Plans
Step 1: Initialized set of query plans, P_0
Step 2: Evaluate Objective 1 (Proximity cost) and Objective 2 (communication cost) for each Query plan.

Obj1: (PC) = $\sum_{i=1} \frac{Ki}{CL}(1-\frac{Ki}{CL})$; // Ki is count of i_{th} server used in query plan, CL is length of query plan//

Obj2:(CC)= Spanning Tree (query plan[]); //to evaluate minimum communication cost among all possible combination of sites for specific query//

Step3: Set a objective function counter i=1;
Define q =N/M; // N is size of population & M no. of objective//
For (J = $1+(i-1)*q$ to J= $i*q$) //Assign fitness for each query plan within subpopulation//
{
$F(x_i^{(J)})=f(x_i^{(J)})$; //$f(x_i^{(J)})$ is objective value & $F(x_i^{(J)})$ is fitness value of i_{th} chromosome in j_{th} subpopulation// }
Step 4: Perform Selection through Tournament selection method on each sub-populations (q), mating pool p_i
Step 5: #if (i=M) then Go to step (6)
#else i =i+1; and Go to step (2)
Step 6: Combine all mating pools from sub-populations :**Mating Pool** = $\bigcup_{i=1}^{M} Pi$ //Pi is mating pool for i_{th} subpopulation//
Step 7: Apply genetic operator on mating pool of query plans,
Apply Crossover with Crossover probability (Pc)
Mutate chromosome with Mutation probability (Pm)
Step 8 : #if (stopping criteria meet), then
Final set of chromosome copy to output (Optimal solutions);
#else Go to Step (2); // to next iteration//

(b) *Example*

In GA, first chromosomes for a population are initialized. In proposed solution, two important aspects are the values used for encoding a chromosome and the sizes of the chromosome. The size of a QEP(chromosome) is equal to the number of relation required in FROM clause of the query and values in a QEP are name of sites of relations. Tables 1 and 2 show relation site matrix and some valid query plans. VEGA initially divides entire population into subpopulations, as 20 chromosomes are in Table 3, initial population (P_0). The number of objective decides the size of subpopulation (q) = N/M, where N is the total number of query plans and M is the number of objectives. Initial population (P_0) consists of 20 query plans, and is now equally divided into subpopulation of 10 query plans in each.

Table 1 Relation site matrix

	Site								
Relation	1	2	3	4	5	6	7	8	9
R_1	1	0	0	1	0	1	0	0	1
R_2	1	0	0	1	1	0	1	0	0
R_3	1	0	1	0	1	0	0	1	0
R_4	1	1	0	1	0	0	1	0	0

Table 2 Valid query plans (chromosomes)

Query plan	Description
[1,1,1,1]	All relations are in site 1
[4,4,3,7]	R_1 and R_2 from site 4, R_3 from site 3 and R_4 from site 7
[6,7,8,2]	R_1, R_2, R_3, R_4 from site no. 6, 7, 8, 2 respectively
[4,4,8,4]	R_1, R_2, R_4 from site 4 and R_3 from site 8

Table 3 Partitioned population, subpopulation 1 (objective 1) and subpopulation 2 (objective 2)

S. no.	Query plans	Fitness value	Comm. cost
1	**[4,4,3,4]**	6/16	32
2	**[1,1,1,1]**	0	0
3	**[4,4,1,2]**	10/16	21
4	**[9,4,8,4]**	10/16	46
5	**[6,1,1,1]**	6/16	20
6	**[9,4,3,4]**	10/16	74
7	**[4,5,5,4]**	8/16	16
8	**[6,5,5,1]**	10/16	36
9	**[1,7,8,4]**	12/16	61
10	**[4,4,8,4]**	6/16	61
11	**[6,4,8,4]**	10/16	70
12	**[1,4,4,1]**	8/16	20
13	**[4,5,5,9]**	12/16	63
14	**[9,5,5,9]**	8/16	47
15	**[6,5,5,9]**	10/16	51
16	**[1,7,5,9]**	12/16	38
17	**[9,7,8,9]**	12/16	30
18	**[6,4,5,4]**	10/16	56
19	**[9,1,8,4]**	12/16	44
20	**[1,1,1,9]**	6/16	14

Table 4 Mating pool sub

Mating pool 1			
S No	Query plans	Fitness value	Comm. cost
1	**[4,4,3,4]**	**6/16**	32
2	**[1,1,1,1]**	**0**	0
10	**[4,4,8,4]**	**6/16**	61
5	**[6,1,1,1]**	**6/16**	20
6	**[9,4,3,4]**	**10/16**	74
7	**[4,5,5,4]**	**8/16**	16
4	**[9,4,8,4]**	**10/16**	46
Mating pool 2			
18	**[6,4,5,4]**	10/16	**56**
12	**[1,4,4,1]**	8/16	**20**
17	**[9,7,8,9]**	12/16	**30**
19	**[9,1,8,4]**	12/16	**44**
14	**[9,5,5,9]**	8/16	**47**
20	**[1,1,1,9]**	6/16	**14**
15	**[6,5,5,9]**	10/16	**51**

Query: ***Select*** *a, b*
From R_1, R_2, R_3, R_4
Where $R_1.a = R_4.t$
and $R_4.p = R_2.x$
and $R_2.x = R_3.n$;

In this query R_1, R_2, R_3, R_4 are required for result retrieval. Relation R_1 is replicated on sites (1, 4, 6, 9), R_2 at sites (1, 4, 5, 7), R_3 at sites (1, 3, 5, 8) and similarly relation R_4 at sites (1, 2, 4, 7) as shown in Table 1 relation site matrix (RSM). Total number of relation required by the query is 4, and the QEP size would be 4. The values into QEP are names of sites for the relation and ordering will be from R_1 to R_4, from left to right.

Selection of the fitter and better chromosomes is achieved parallel in VEGA, as both subpopulation are treated differently with different heuristics, as first subpopulation is based on the heuristic 1 and second on the heuristic 2. In Table 3, query plans with lower fitness value are inserted into the mating pool from subpopulation 1, similarly from the subpopulation 2, query plans with lower communication cost are inserted into mating pool.

Now for implementing the crossover and mutation, combine all mating pool of different subpopulations. Table 4 mating pool consists of fitter and better QEPs, combine mating pool 1 and pool 2. Next, apply crossover with P_c in [0.5, 0.9]. In graph section, the effect of different crossover probability (P_c) values on generation

Table 5 New population

S No	Query plans	Fitness value	Comm. cost
4	[9,4,8,4]	10/16	46
7	[4,5,5,4]	8/16	16
2	[1,1,1,1]	0	0
1	[4,4,3,4]	6/16	32
5	[6,1,1,1]	6/16	20
12	[1,4,4,1]	8/16	20
14	[9,5,5,9]	8/16	47
20	[1,1,1,9]	6/16	14
N1	**[4,4,5,4]**	**6/16**	**14**
N2	**[6,5,5,4]**	**8/16**	**25**
N3	**[6,4,5,9]**	**10/16**	**41**
N4	**[6,5,8,4]**	**10/16**	**42**
N5	**[4,4,5,9]**	**8/16**	**16**
N6	**[9,4,5,9]**	**8/16**	**16**
N7	**[6,5,3,4]**	**10/16**	**40**
N8	**[9,4,3,4]**	**8/16**	**18**
N9	**[9,1,5,4]**	**10/16**	**41**
N10	**[9,4,5,4]**	**8/16**	**23**
N11	**[4,5,8,4]**	**8/16**	**20**
N12	**[9,1,3,4]**	**10/16**	**42**

of new population is shown. Mutation required in the QEP (chromosomes) according to P_m value, for VEGA P_m, is [0.0, 0.2]. In Table 5, new population consists of new set of QEPs for initial population given in Table 4.

(c) *Graphs*

The comparative analysis of optimization performance is according to parameters, e.g. Number of generations, Average query cost and Top-K query of VEGA; aggregation-based genetic algorithm is shown in the following graphs. The experimental setup includes simulation on genetic algorithm toolkit in MATLAB with modelled cost functions and pool of QEP given relation site matrix. For the comparative experimentation, different crossover schemes are implemented: single point crossover (SPC), double point crossover (DBC) and uniform point crossover (UPC). UPC converged more consistently as compared to other schemes for all set of parameters. In Figs. 2 and 3, the comparative results are shown. The experimental analysis states that VEGA's convergence is better than any multi-objective aggregation-based genetic algorithm for different values of crossover and mutation probability, P_c and P_m.

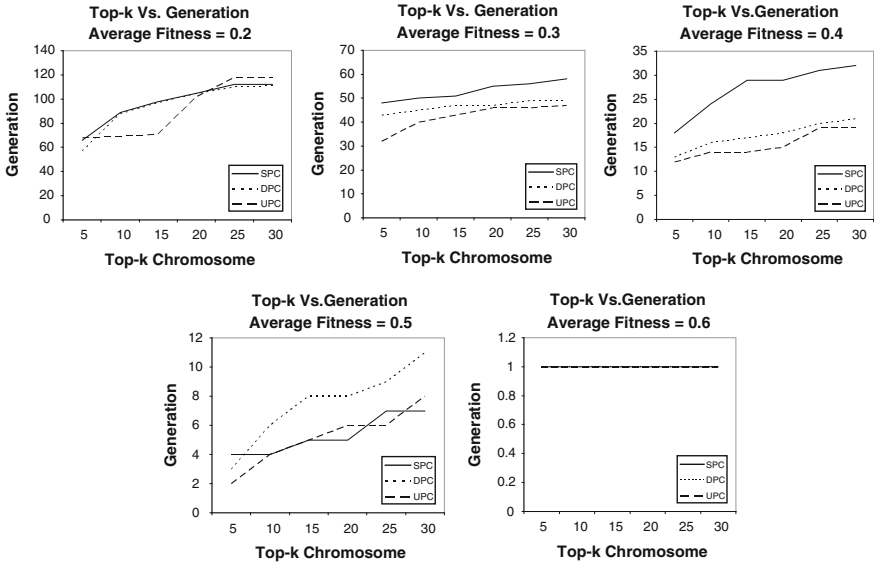

Fig. 2 Top-K versus generation for fixed average fitness

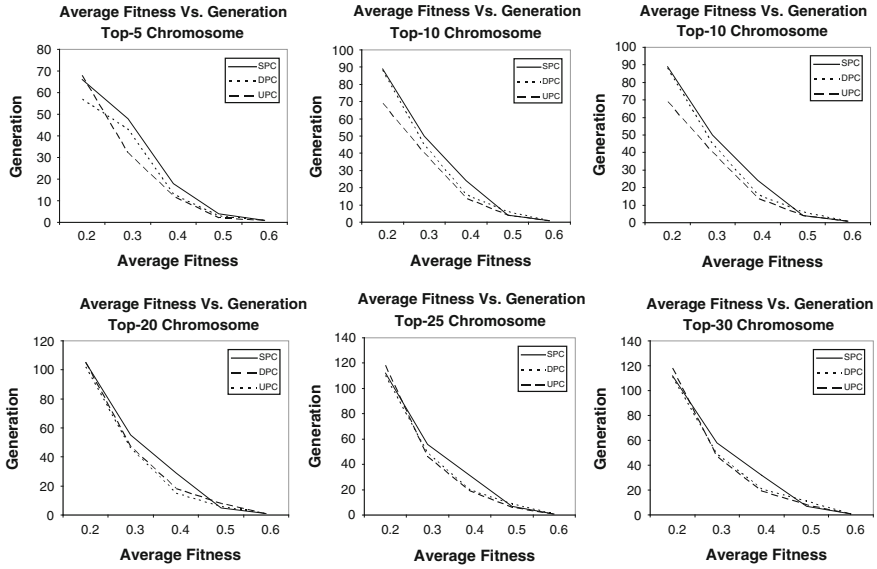

Fig. 3 Average fitness versus generation for fixed Top-K

Conclusion

Distributed database system is one of the most reliable processing environments; logical data is kept in multiple sites. The users at local sites can work independently as well as communicate with other sites to retrieve data for answering global queries; setup is defined as distributed query processing (DQP). A distributed query processing strategy aims to reduce the overall cost of query processing. The query optimizer primarily minimizes the time required for result retrieval for a user query. Genetic algorithm is nature-based optimization technique which has also been widely used in solving complex real-life optimization problems. The strength of GA as a meta-heuristic comes from its ability to combine the good features from several solutions to create new and better solutions over generations. Schaffer proposed a multi-objective genetic algorithm to accommodate the different types of fitness function, vector valued fitness values, called vector evaluated genetic algorithm. Initial steps are similar to single-objective GA, while selection is modified, so that at proportional selection according to each of the objectives is achieved. This selection approach justifies the importance of each of the design objectives of optimization problem, and in query optimization query plans are selected for evolution in proportional contribution of each of the objective populations. The performance trade-offs are discussed result section, and the comparative graphs are drawn between various parameters, e.g. number of generation (evolution) and average query cost.

References

1. Bernstein, P.A., Goodman, N., Reeve, C.L, Rothnie, J.B., Wong, E.: Query processing in a system for distributed database. ACM Trans. Database Syst. **4**(602–625) (1981)
2. Chu, W., Hurley, P.: Optimal query processing for distributed database systems. IEEE TC **C-31**(835–850) (1982)
3. Chang, C.C., Yu, C.T.: Distributed query processing. ACM Comput. Surv. **16**(4), 399–433 (1984)
4. Ceri, S., Pelagati, G.: Distributed Database: Principles and Systems. McGraw Hill (1984)
5. Gregory, M.: Performance issues in distributed query processing. IEEE Trans. Parallel Distrib. Syst. **4**(8) (1993)
6. Kossmann, D.: The State of the art in distributed query processing. ACM Comput. Surv. (2000)
7. Chang, J.M.: A heuristic approach to distributed query processing. In: Proceedings of VLDB (1982)
8. Jarke, M., Koch, J.: Query optimization in database systems. ACM Comput. Surv. (1984)
9. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evol. Comput. **3**(1), 1–16 (1995)
10. Coello, C.A.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl. Inf. Syst. (1999)
11. Ishibuchi, H., Narukawa, K.: Comparison of evolutionary multi-objective optimization with reference solution-based single-objective approach. In: Proceedings of GECCO-2005, USA, pp. 787–794 (2005)

12. Fleming, P., Wang, R., Purshouse, R., Fleming, P.: Local preference-inspired co-evolutionary algorithms, In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, vol. 3, no. 1, pp. 513–520 (2012)
13. Vijay Kumar, T.V., Singh, V., Verma, A.K.: Int. J. Comput. Theory Eng. **3**(1) (1793–8201) (2011)
14. Panicker, S., Vijay Kumar, T.V.: Distributed query plan generation using multiobjective genetic algorithm. In: ICICA (2011)
15. Goldberg, D., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Found. Genet. Algorithms (69–93) (1991)
16. Epstein, S.R., Wang, M.E.: Distributed query processing in relational databases system. In: Proceedings of ACM SIGMOD (1978)
17. Kambayashi, Y.S., Yoshikawa, M.: Query processing for distributed databases using generalized semi-joins. In: International Conference of Management of Data in ACM SIGMOD, pp. 151–160 (1982)
18. Bodorik, P., Riordon, J.S.: Distributed query processing optimization objectives. In: Proceedings of the IEEE Fourth ICDE, LA CA, pp. 320–329 (1988)
19. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
20. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1998)
21. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley (2001)
22. Deb, K., Goldberg, D.E.: An investigation of niche and species formation in genetic function optimization. In: Proceedings of the Third ICGA, pp. 1–10 (1990)
23. Deb, K.: Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evol. Comput. **7**(3), 205–230 (1999)
24. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of ICGA, Hillsdale, pp. 93–100 (1987)
25. Zitzler, E., Deb, K., Thiele, L.: Comparison of multi-objective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)
26. Deb, K., Agrawal, S.: Understanding interactions among genetic algorithm parameters. Found. Genet. Algorithms **V**, 265–286 (1998)
27. Yu, C.T., Guh, K.C., Chen, A.L.P.: An integrated algorithm for distributed query processing. In: IFIP Conference on Distributed Processing, Amsterdam (1987)