

Research on SQLite Database Query Optimization Based on Improved PSO Algorithm

Aite Zhao¹, Zhiqiang Wei¹ and Yongquan Yang^{1,*}

¹ Ocean University of China, 266100, Qingdao, Shandong, China
tiddyzhao@hotmail.com, weizhiqiang@ouc.edu.cn, i@yangyongquan.com

Abstract

In recent years, with the development of the information industry, we ushered in the age of big data, more and more Internet and mobile products are popping up. According to the mobile device is portable, real-time etc, the development of mobile phone system become the focus of scientific and technological development, but the research on its local database is relatively small. This paper introduced a query optimization method based on improved Particle Swarm Optimization algorithm (PSO) for SQLite database on Android platform. This method improves the original PSO, and put the database transaction into the Particle Swarm Optimization algorithm, and should be used to join query. It improves the speed of complex query, and optimizes the query on SQLite database. Experimental results show that this method is an effective way to optimize the SQLite database query, and also can be used in the Android platform.

Keywords: query optimization, Particle Swarm Optimization algorithm, SQLite, Android, database transaction, join query

1. Introduction

Android operating system is a universal operating system, and it has become one of the world's most popular mobile platforms. It is easy to use in single binary cell phones, tablet PCs, and other devices. The number of the smart phone and tablet PC that use the Android operating system has been more than 1 billion [1]. Moreover, the local database of Android system is convenient. SQLite [2] is an embedded and lightweight database on Android operating system. SQLite supports SQL syntax of standard relational database for complex queries. You only need to define the Create or Update SQL statement, and then SQLite can automatically manage your Android platform. Most database query optimization algorithms are working on PC -side optimization, ignoring the lightweight SQLite database optimization. This paper has improved Particle Swarm Optimization algorithm with the Android platform SQLite database, and constructed a model which is suitable for Android platform.

Particle Swarm Optimization (PSO) is an algorithm developed gradually in recent years. PSO is based on the social – psychological principle and is a randomized algorithm based on the crowd. Unlike the evolutionary algorithm, PSO does not choose, typically, all members of the group exist from start to end. With the passage of time and iterative improvement of the quality of solutions, results of their interaction will be improved [3], [4]. This paper presents an algorithm that can be used in the Android platform, based on improved PSO algorithm. Using this algorithm to create model on the Android platform, and use the more complex join query as an example to research. First, we should build a syntax tree for the join query, and then use the tree coding to transform the tree and generates the corresponding connected graph. The next step is to calculate the fitness and take out the optimal solution according to the fitness, and then update the position and velocity of the particle. These steps should be loop iteratively until the query finished.

This model uses the Android 4.4 Version as the operating system, and uses Office Automation (OA) application to do simulation experiment for the improved PSO algorithm.

2. Related Work

There are a lot of algorithms for query optimization. The main types of the main are dynamic programming algorithm, swarm intelligence algorithm, genetic algorithm, ant colony optimization algorithm, Simulate Anneal Arithmetic and hybrid optimization algorithm. The following is mainly about genetic algorithm and ant colony optimization algorithm (ACO).

2.1. Genetic Algorithm

Genetic algorithm is derived from Darwin's theory of evolution. First, the problem is abstracted as a population, then code individual of this population, and the next step is to calculate individual fitness, and extract individuals which have high fitness as an initial population. Then Individuals in this population will do genetic operation (crossover and selection and mutation), loop to do this thing and produce more strong offspring, and finally determine the optimal solution [5]. In this algorithm, the process of calculating fitness is similar to PSO. They are based on the study of the swarm intelligence algorithm.

2.2. Ant Colony Optimization Algorithm

ACO is derived from the foraging behavior of ants. Many Ants will release a kind of secretion called information pigment in began foraging, the amount of information pigment decide the distance between paths, the higher the concentration of information pigment is, the more ants who select this path will be attracted. Information pigment of paths also will volatile with the passage of time. Information pigment of shortest path that can reach food will be increasingly more, and information pigment of path cannot reach food will increasingly less, The optimal solution can be determined by the number of information pigment [6-7]. However, when ants choose the same path after the beginning of the food, then the information pigment on that path will become more, so that local optimal solution will be obtained possibly in the later stage. The distinction between information pigments can be studied, they are the number after find the food, and the number hasn't found the food.

2.3. Particle Swarm Algorithm

Also called Particle Swarm algorithm of particle swarm optimization algorithm, from the birds of prey, PSO can be regarded as a flock, are determined by the loop iteration of Particle Swarm Optimization. The process is similar to genetic algorithm, but not cross in the iterative process, selection and mutation genetic manipulation. First to initial of a particles group, in it, each particle has a adapted value, on each particle for adapted value calculation, find optimal solution, into iterative process, then according to adapted value on optimal solutions for adjustment, update location and speed of birds, if reached maximum number of iterations or minimum errors, iterative process should be ended, determine optimal solutions, otherwise, continues to calculate adapted value for next iteration [8-9].

3. Database Query Optimization Algorithm Based on Particle Swarm Optimization Algorithm

Now I will introduce the main method of the optimization algorithm. It contains two parts and some important steps.

3.1 Particle Swarm Optimization Algorithm

Suppose size of a particle is n and the space dimension is d , then the current position of the i th particles is:

$$X_i = (X_{i1}, X_{i2}, X_{i3}, X_{i4}, X_{i5}, \dots, X_{id}), 0 \leq i \leq n.$$

The velocity vector of the i th particle:

$$V_i = (V_{i1}, V_{i2}, V_{i3}, V_{i4}, V_{i5}, \dots, V_{id}), 0 \leq i \leq n.$$

Search for the individual optimal solution ($pBest$):

$$P_i = (P_{i1}, P_{i2}, P_{i3}, P_{i4}, P_{i5}, \dots, P_{id}), 0 \leq i \leq n.$$

Search for the global optimal solution ($gBest$):

$$P_g = (P_{g1}, P_{g2}, P_{g3}, P_{g4}, P_{g5}, \dots, P_{gd}).$$

The formula of updating velocity and position of each particle is:

$$V_{id} = W * V_{id} + c_1 r_1 (p_{id} - X_{id}) + c_2 r_2 (P_{gd} - X_{id}).$$

$$X_{id} = X_{id} + V_{id}.$$

Parameters: W in the formula is inertia weight; you can use the Linearly Decreasing Weight (LDW) strategy to define it. Parameter c_1 and c_2 represent learn factors, usually, we make $c_1 = c_2 = 2$ to ensure its efficiency. Parameter r_1 and r_2 are average random numbers that in the range of 1 to 5. $V_{id} \in [-V_{max}, V_{max}]$, users can set the maximum value and minimum value of the velocity, but if the velocity was too big or too small, it will make the results deviate from the optimal solution [10-11].

3.2 Sqlite Database Query Optimization Algorithm

SQLite query optimization mainly includes the following steps:

0) Uses the concept of database transaction to do optimization for database query process. SQLite database embedded in the Android system, database operations are timed to commit the transaction, this means that it commits once in a while, increased the time consumption for reading and writing file, so add transaction operation helps to improve the efficiency of database query [12]. Using function `begin Transaction ()` to manually open the transaction is preparation of database query.

1) Initializes the particle swarm, and then calculates each particle's position, velocity and fitness, and forms an orderly connection string.

2) Finds (updates) individual best solution ($pBest$).

3) Finds (updates) the global best solution ($gBest$).

4) Updates particles' position and velocity, and select connected relation and put them into the collection, finally orderly string.

5) To determine whether the number of iterations reaches the maximum, if you reach the maximum number of iterations, you end a query, otherwise continue with c), d), e), and update the optimal solution.

6) Obtains the optimal query plan.

7) End query.

8) Uses the `setTransactionSuccessful ()` function tags query was successful.

9) Uses the `endTransaction ()` function to commit the transaction.

The detailed process of database query optimization is shown in the following Figure1.

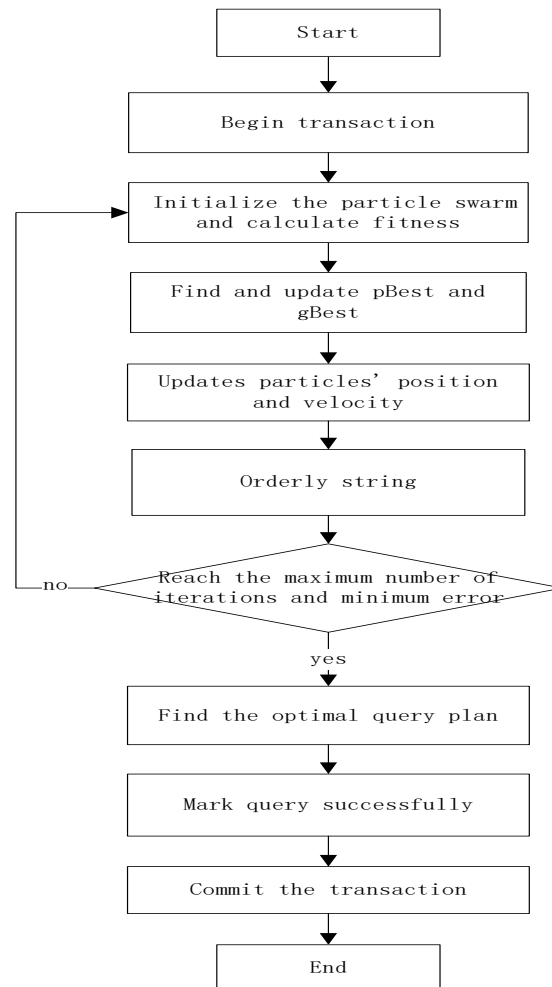


Figure 1. Flow Chart of Database Query Optimization Algorithm

4. Database Query Optimization Design

This paper is based on improved particle swarm algorithm with more complex join query as an example for query optimization. Office Automation (OA) is an application on the basis of the Android operating system, its database SQLite contains many tables; now only take 4 tables (files, position, admin, singlefile) to do join query. The steps are as follows:

4.1 Build a Query Connected Graph

First of all, we should generate the connected graph according to the connection query statement. The query statement is: select files._id, fileName, files.filesNo, fileName, pos, name, time from files inner join singlefile on files.filesNo=singlefile.filesNo inner join position on singlefile.filesNo=position.filesNo inner join admin on position.filesNo=admin.filesNo. According to the above connected graph, the connected graph is $G(V, E)$, V is the node, and E is the edge, then the connected graph is obtained as follows:

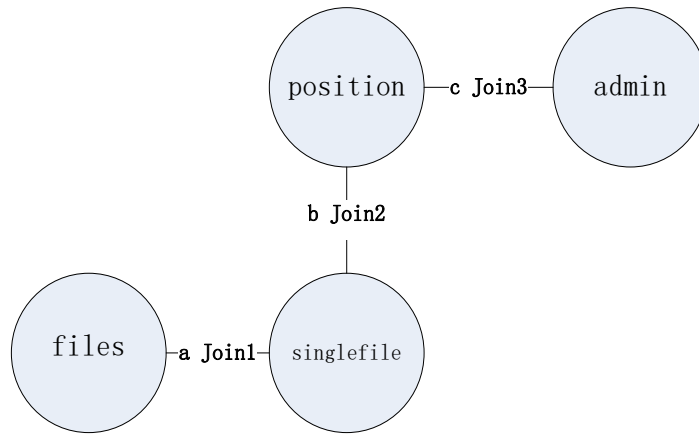


Figure 2. Join Query Connected Graph

Remark 1. In Figure2, a, b, c, d are connection properties, join1, join 2, join 3, join 4 are Connection identifications. Constructing the syntax tree of the query statement is according to the node of the connection.

4.2 Create the Initial Population, Build a Binary Tree

According to the properties and identifies of the graph, it is necessary to encode the query statement, due to the simple and easy method to operate, and avoid the situation that binary encoding may result in illegal solution, In this paper, we use binary tree coding, the generated syntax tree are as follows:

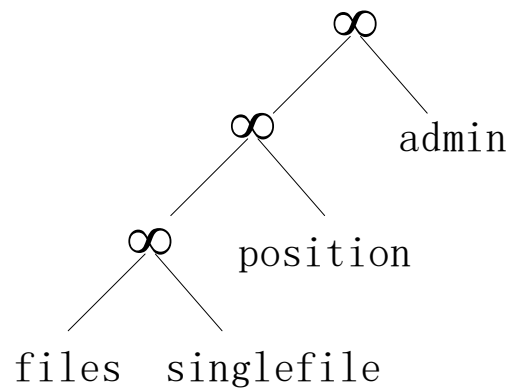


Figure 3. Join Query Syntax Tree

Remark 2. According to the syntax tree shown in Figure3, first encoding the binary tree by DLR, the value of leaf nodes is 1, the value of non leaf nodes is 0, the corresponding code is (0,0,0,1,2,3,4), due to the connection operations are in non leaf nodes, so 0 values are in the top, code can be simplified to (1,2,3,4). Coding sequence is the initial population. Each code number in the sequence is the position of each particle in the mass, after the DLR, velocity of particles can be generated.

4.3 Calculate Fitness

Particles in the process of finding the optimal solution require fitness to judge. The computation of fitness needs to design fitness function, and fitness function also directly affects the convergence rate and optimal solution of the particle.

If there are n relations in the query, the internal node number of its syntax tree is t_i , the cost of the query is the sum of the nodes of syntax tree, and the mathematics is as follows.

$$\text{consume}(t) = \sum_{i=1}^{n-1} \text{consume}(t_i)$$

Calculating the fitness first should determine the appropriate cost function $\text{consume}(P)$, take the reciprocal, is fitness function of particles.

$$F(P) = 1/\text{consume}(P)$$

At first, the fitness of all particles in the particle swarm should be calculated, and the particles with high fitness are taken. The greater the consumption is, the smaller the particle's fitness is.

4.4 Update Particles' Position and Velocity

First of all, we rank all the particles according to their fitness value from high to low, particle which has the highest adaptive value will enter the next round of iteration, and other particles will update the position and velocity, In the process of updating, to set the minimum and maximum values of the particle location. When encounter a duplicate location value, plus one, until the location is available.

When the maximum number of iterations is reached, the loop is over, and the optimal query plan is found. Then the transaction is submitted manually and the query is completed.

5. Algorithm Test and Result Analysis

After optimizing the application of office automation by using the improved PSO, query statement becomes: `select files._id, fileName, files.filesNo, fileName, pos, name, time from files inner join position on files.filesNo=position.filesNo inner join admin on position.filesNo=admin.filesNo inner join singlefile on singlefile.filesNo=admin.filesNo`. Doing join query for the 4 tables files, position, admin, singlefile, using the function `System.nanoTime()` to calculate the running time difference before and after the query, and then print to the logcat console. Print results are shown in Figure4:

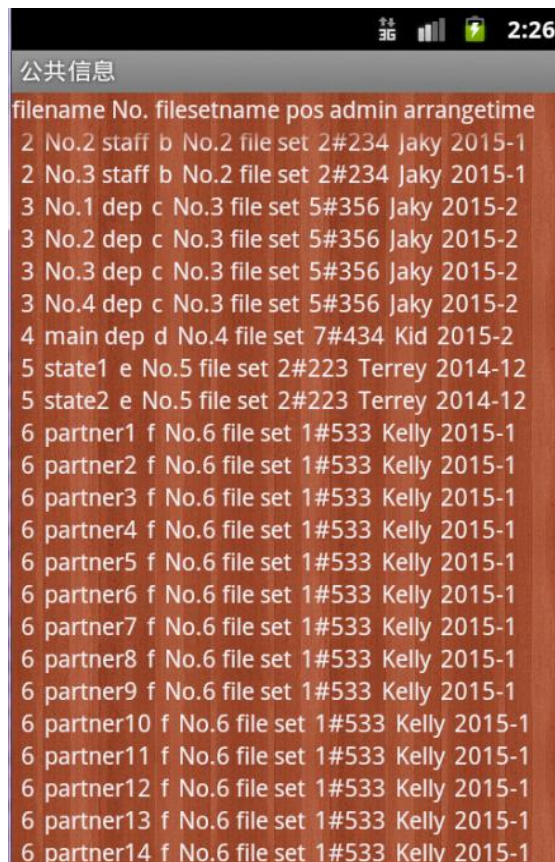
PID	TID	Application	Tag	Text
3763	3763	com.tt.oe	before optimization:	525852749
3763	3763	com.tt.oe	after optimization:	14805400
3763	3763	com.tt.oe	before optimization:	23207852
3763	3763	com.tt.oe	after optimization:	11834899
3763	3763	com.tt.oe	before optimization:	48660223
3763	3763	com.tt.oe	after optimization:	24728200
3763	3763	com.tt.oe	before optimization:	30252090
3763	3763	com.tt.oe	after optimization:	12180453
3763	3763	com.tt.oe	before optimization:	23387649
3763	3763	com.tt.oe	after optimization:	11901927

Figure 4. Query Time Comparison Before and After Optimization

Remark 3. By computing the average of the results of the above five groups, the optimized query time is 0.1151819368s faster than no-optimization query. The amount of

data in the four tables is less, the maximum number of records in the four tables is only 39, and the more datas in SQLite database is, the more obvious the effect will be.

Query result is shown in the following figure:



filename	No.	filesetname	pos	admin	arrangetime
2	No.2	staff	b	No.2 file set 2#234	Jaky 2015-1
2	No.3	staff	b	No.2 file set 2#234	Jaky 2015-1
3	No.1	dep	c	No.3 file set 5#356	Jaky 2015-2
3	No.2	dep	c	No.3 file set 5#356	Jaky 2015-2
3	No.3	dep	c	No.3 file set 5#356	Jaky 2015-2
3	No.4	dep	c	No.3 file set 5#356	Jaky 2015-2
4	main	dep	d	No.4 file set 7#434	Kid 2015-2
5	state1	e	No.5	file set 2#223	Terrey 2014-12
5	state2	e	No.5	file set 2#223	Terrey 2014-12
6	partner1	f	No.6	file set 1#533	Kelly 2015-1
6	partner2	f	No.6	file set 1#533	Kelly 2015-1
6	partner3	f	No.6	file set 1#533	Kelly 2015-1
6	partner4	f	No.6	file set 1#533	Kelly 2015-1
6	partner5	f	No.6	file set 1#533	Kelly 2015-1
6	partner6	f	No.6	file set 1#533	Kelly 2015-1
6	partner7	f	No.6	file set 1#533	Kelly 2015-1
6	partner8	f	No.6	file set 1#533	Kelly 2015-1
6	partner9	f	No.6	file set 1#533	Kelly 2015-1
6	partner10	f	No.6	file set 1#533	Kelly 2015-1
6	partner11	f	No.6	file set 1#533	Kelly 2015-1
6	partner12	f	No.6	file set 1#533	Kelly 2015-1
6	partner13	f	No.6	file set 1#533	Kelly 2015-1
6	partner14	f	No.6	file set 1#533	Kelly 2015-1

Figure 5. Join Query Results

6. Conclusion

Above, through the analysis and research on PSO algorithm, enhanced efficiency of join query statement, and the improved PSO algorithm is applied to Android operating system, optimized the SQLite database, and combine the database transaction and particle swarm optimization algorithm, it can improve the performance of SQLite database. With the constant exploration of the PSO algorithm, I hope the algorithm will be applied in more and more fields.

Acknowledgement

This paper is supported by the Fundamental Research Funds for the Central Universities (No.201413065), key Science and Technology Program of Shandong province (No. 2014GGX101005), and Qingdao strategic emerging industry development plan (No. 13-4-1-45-hy).

References

- [1] The Android Story, "Android, the world's most popular mobile platform", <http://www.android.com/historyhttps://developer.android.com/about/index.html>, (2015).
- [2] M. Owens and G. Allen, "The definitive guide to SQLite", Berkeley: A press, (2006).

- [3] J. Sun, W. Fang and X. J. Xu, "Quantum-behaved particle swarm optimization: analysis of the individual particle's behavior and parameter selection", *Evolutionary Computation*, vol. 20. No. 3, (2012), pp. 349.
- [4] D. S. L. Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems", *Expert Systems with Applications*, vol. 37, (2010), pp. 1676-1683.
- [5] K. Deb, A. Pratap and S. Agarwal, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2. (2002), pp. 182-197.
- [6] M. Dorigo, "Special section on ant colony optimization", *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 4, (2002), pp. 317-319.
- [7] L. J. Ke, Q. F. Zhang and R. Battiti, "MOEA/D-ACO: A multi-objective evolutionary algorithm using decomposition and ant colony", *IEEE Transactions on Systems Man and Cybernetics Part A-Systems and Human*, no. 99, (2013), pp. 1-15.
- [8] G. Lin, "Optimization of database query application research based on particle swarm algorithm", vol. 29, no. 3, (2012), pp. 974-975.
- [9] C. Guo, L. Zhu and X. Li, "Multi-join Query Optimization Method Based on Ant Colony Algorithm", *Computer Engineering*, vol. 35, no. 10, (2009), pp. 173-176.
- [10] C. T. Man and G. M. Sheng, "An Improved Algorithm Based on Cooperative Particle Swarm Optimization", *Journal of Harbin University of Science & Technology*, (2010).
- [11] G. H. Guo and Z. G. A. Wang, "Modified Particle Swarm Optimization", *Journal of Harbin University of Science & Technology*, (2010).
- [12] G. Allen and M. Owens, "The definitive guide to SQLite", American, A press, (2010), pp. 193-144.