1 The purpose of SAIO 3

2 User documentation 5
  2.1 Setup SAIO .................................................. 5
  2.2 GUC variables ............................................... 5

3 How to contribute 7
  3.1 Exploring SAIO ............................................. 7
  3.2 Bug reports .................................................. 7
  3.3 Extending SAIO code ....................................... 7
  3.4 Proposing new algorithms of move generating ............... 7

4 Developer’s documentation 9
  4.1 Files overview ............................................. 9

5 For researchers 11
Contents:
The purpose of SAIO

The purpose of SAIO is to show how to write PostgreSQL extensions which make use of JOIN_SEARCH_HOOK and to stimulate the search of a better and cleaner non-exhaustive join order optimizer.

In the past SAIO was supposed to replace GEQO, however the latest benchmarks showed that GEQO is able to produce better results for most of used queries.

A recognized case where SAIO generally beats GEQO is optimizing queries with several RIGHT JOIN relations.
2.1 Setup SAIO

To use SAIO, you will need the PostgreSQL development headers. Compile and install with:

```
$ make
$ sudo make install
```

After that log in to your PostgreSQL server with a superuser account and issue:

```
=# LOAD 'saio';
=# SET saio_threshold TO 10;
```

By default all queries with the number of FROM elements exceeding saio_threshold will be planned using SAIO. To disable SAIO use:

```
=# SET saio TO 'false';
```

Beware, if the module has been compiled against a server with assertion checking enabled, it will run extremely slowly and it will write debugging information to the /tmp directory.

2.2 GUC variables

The GUC variables the module defines are:

2.2.1 saio

default: true

A boolean that enables or disables using SAIO for planning. If true, all queries with number of joins exceeding saio_threshold will be planned with SAIO, if false none.

2.2.2 saio_threshold

range: 0 to INT_MAX
default: 14

Determines the threshold of FROM items beyond which SAIO is used.

For smaller queries it is suitable to optimize join order with PostgreSQL default exhaustive optimizer.
2.2.3 saio_seed

range: 0.0 to 1.0
default: 0.0

A floating point seed for the random numbers generator. If saio_seed is set to 0.0, the algorithm will randomize the seed with time(NULL). For other values, it will use the fixed seed provided.

2.2.4 saio_equilibrium_factor

range: 1 to INT_MAX
default: 16

Scaling factor for the query size, determining the number of loops before equilibrium is reached.
Higher equilibrium_factor gives SAIO possibility to visit more states but increases the computation time.

2.2.5 saio_initial_temperature_factor

range: 0.0 to 10.0
default: 2.0

Factor determining the initial temperature of the system.
Higher saio_initial_temperature_factor gives SAIO possibility to visit more states but increases the computation time.

2.2.6 saio_temperature_reduction_factor

range: 0.0 to 1.0
default: 0.9

Factor determining how much the temperature is reduced each time equilibrium is reached.
Lower saio_temperature_reduction_factor gives SAIO possibility to visit more states but greatly increases the computation time.

2.2.7 saio_moves_before_frozen

range: 1 to INT_MAX
default: 4

How many moves without a state change are considered a freezing condition.
How to contribute

You can contribute to make this project better in various ways.

3.1 Exploring SAIO

You can extend the current benchmarks to expose cases where SAIO works better than GEQO.

3.2 Bug reports

If you encounter any issues with the module, don’t hesitate to send a BUG report to https://github.com/parkag/saio/issues.

3.3 Extending SAIO code

If you see that the developers of SAIO missed something and a part of the extension could be improved, don’t hesitate and create a pull request!

Since SAIO is CPU bounded, one could try to develop a concurrent version of the extension to improve the running speed.

3.4 Proposing new algorithms of move generating

If you find a great algorithm of moving through the solution space of SAIO, consult with the developers of SAIO or just implement it and submit a pull request!
4.1 Files overview

4.1.1 doc

Contains this documentation. To build it use:

```
make html
```

in the doc directory

4.1.2 scripts

Contains various snippets useful for the purpose of debugging SAIO.

4.1.3 src

Contains SAIO headers and source files. The code is well documented so there is no point in repeating the comments here.

```
saio_main.c

Contains the main loop of the simulated annealing algorithm. In pseudocode:

do {
    do {
        new_state = random_move()
        if (acceptable(new_state))
            state = new_state
    } while (!equilibrium())
    reduce_temperature()
}while (!frozen())
return state
```
saio.c

Defines GUC variables and manages JOIN_SEARCH_HOOK usage.

saio_trees.c

Contains operations on tree-like structures which represent join relations between tables.

saio_util.c

Contains functions that validate joins and generate random numbers.

saio_recalc.c

The implementation of SAIO_recalc algorithm. The algorithm exposes such methods:

```c
SaioAlgorithm algorithm = {
    .step = saio_recalc_step,
    .initialize = saio_recalc_initialize,
    .finalize = saio_recalc_finalize
};
```

4.1.4 test

Contains regression tests for SAIO. To run tests go to the top directory and use:

```bash
make installcheck
```

The tests will be run in alphabetical order.

For a test to pass the output generated from running an sql script from the sql subdirectory must be identical to the corresponding .out file from the expected subdirectory.

It is also possible to generate test coverage reports through gcov and lcov. To use this you need to have your PostgreSQL installed from the source code with `--with-coverage` flag enabled.

```bash
./configure --with-coverage
```

To generate coverage reports for SAIO run:

```bash
make installcheck
make coverage
```

This will generate coverage directory with the coverage report. The test coverage should be ideally kept in the range of 95%-100%.
For researchers

Some materials which may be helpful to start research on heuristics in join order optimizing.

2. PGCon2010 presentation on an earlier version of SAIO by Jan Urbański: presentation
3. Comparison of the current SAIO version with GEQO in PostgreSQL 9.4 by Grzegorz Parka: benchmarks

Supplementary material:

1. Experimental comparison of Iterative Improvement, Genetic Algorithms and Simulated Annealing for query optimization article