

Reactive Search: Quadratic Assignment

User Manual

Version 3.0— November 15, 2007



Reactive Search: Learning on the Job

Reactive Search is a robust and efficient method for solving difficult optimization problems. The word “reactive” hints at a ready response to events *while* alternative solutions are tested. Its strength lies in the introduction of high-level skills often associated to the human brain, such as learning from the past experience, learning on the job, rapid analysis of alternatives, ability to cope with incomplete information, quick adaptation to new situations and events.

The main features of the Reactive Search techniques are:

Learning on the job Real-world problems have a rich structure. While many alternative solutions are tested in the exploration of a search space, patterns and regularities appear. The human brain quickly learns and drives future decisions based on previous observations. This is the main inspiration source for inserting online *machine learning* techniques into the optimization engine of Reactive Search.

Rapid generation and analysis of many alternatives Often, to solve a problem one searches among a large number of alternatives, each requiring the analysis of *what-if* scenarios. The search speed is improved if alternatives are generated in a strategic manner, so that different solutions are chained along a trajectory in the search space *exploring* wide areas and rapidly *exploiting* the most promising solutions.

Flexible decision support Crucial decisions depend on factors and priorities which are not always easy to describe before starting the solution process. Feedback from the user in the preliminary exploration phase can be incorporated so that a better tuning of the final solutions takes the end user preferences into account.

Diversity of solutions The final decision is up to *you*, not the machine. The reason is that not all qualitative factors of a problem can be encoded into a computer program. Having a set of diverse near-best alternatives is often crucial for the decision maker.

Anytime solutions You decide when to stop searching. A first complete solution is generated rapidly, better and better ones are produced in the following search phases. The more you run, the bigger the possibility to identify excellent solutions, but if you want a solution fast you are going to get it!

The Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) models the real-world task of finding the optimal location for a set of interdependent facilities to minimize the cost of moving commodities between them.

A set of n facilities and a set of n locations are given; a distance is specified for each pair of locations and a *weight* or *flow* is given for every pair of facilities. Such weight represents the amount of supplies to be transported. The request is to assign all facilities to different locations with the goal of minimizing the total cost of moving supplies between facilities; the cost is given by the sum of the distances multiplied by the corresponding flows.

In mathematical terms, the problem data are:

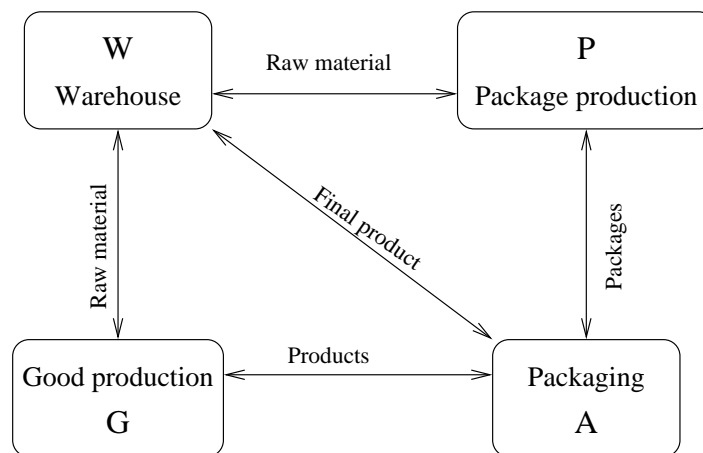
- the number n of facilities and locations;
- $\text{distance}(i, j)$ between locations i and j
- $\text{weight}(i, j)$ describing the flow between facilities i and j .

Note that the number of locations is equal to the number of facilities, so that the problem's outcome must be a *permutation* $(\text{loc}[1], \dots, \text{loc}[n])$ where $\text{loc}[i]$ is the chosen location of facility i . The objective function to minimize is

$$\sum_{i < j} \text{weight}(i, j) \times \text{distance}(\text{loc}[i], \text{loc}[j]).$$

Two steps to solve it!

Let us assume that you need to manage a production system where goods are produced at plant G, and packaged at plant A with packages produced at plant P; raw materials and final products are stored in warehouse W. The following picture describes the main flows between plants:



The warehouse provides raw materials for good and package production; goods and packages are moved to the packaging plant, and the final packages are moved back to the warehouse. Other flows are possible: for example, goods and packages in excess may be moved to the warehouse for later use. Flows are represented by the following table:

	W	G	P	A
W	0	10	8	25
G	10	0	1	9
P	8	1	0	8
A	25	9	8	0

The four locations are identified with numbers from 1 to 4; their mutual distances are shown in the following table:

	1	2	3	4
1	0	5	2	4
2	5	0	3	1
3	2	3	0	4
4	4	1	4	0

Step 1: Encode the problem

Open a new file, call it `test.dat`, with the following content:

```
4
0 5 2 4
5 0 3 1
2 3 0 4
4 1 4 0

0 10 8 25
10 0 1 9
8 1 0 8
25 9 8 0
```

Note that the first number represents the number n of facilities and locations, the first matrix contains all mutual distances $\text{distance}(i, j)$ between locations, the second matrix contains the flows $\text{weight}(i, j)$ between facilities.

Step 2: Call the optimizer

The optimizer is invoked by writing the problem file name as argument, followed by the execution time in seconds; for example, the following will execute optimization for a second:

```
./rts-qap test.dat 1
```

The program outputs the solution's improvement and the best solution found:

```
Problem file test.dat open.
Solver initialization complete.
Running Reactive Search for 1 seconds...
  Found better objective value 332 at step 1
  Found better objective value 330 at step 2

Search for problem test.dat completed:
  Iterations 147379
  Best objective value 330

Locations:
  Facility 1 at location 3
  Facility 2 at location 1
  Facility 3 at location 2
  Facility 4 at location 4
```

Program reference

Program invocation

The `rs-qap` program offers the following command-line syntax:

```
./rs-qap problem-filename time [seed]
```

where

problem-filename

is the name of the text file containing the problem encoding as described in the following section;

time

is the duration (in seconds) of the Reactive Search optimization;

seed

is the optional random number generator seed; when omitted, 999 is assumed as default.

Problem definition file

The problem file contains the number n of locations and plants, followed by the n^2 integer entries of the distance matrix and the n^2 entries of the flow matrix. Both matrices must be symmetric, see the `test.dat` example presented before.

Program output

The program writes all of its messages to the console. In particular, at the end of the execution the following data will be printed:

Iterations

Total number of iterations executed;

Best objective value

The best value of the objective function

Locations

List of location assignments for facilities (the problem's best found solution).

List of files

The package contains the following files:

`RS-QAP-manual.pdf`

This file.

`rs-qap` or `rs-qap.exe`

The binary application.

`test.dat`

Data file for a test problem.