Statistical Computing in R
Report

Implementation of NMMSO in R

Submitted by

425699 Daniel Carriola
425599 Jannik Hoffjann

Under the guidance of
Dr. Mike Preuss

Information Systems and Statistics
ERCIS
Münster - NRW - Germany

Winter Semester 2015/16

Contents

List of Figures

1 Introduction

In the recent years R has become the statistical programming language of choice for many scientist. The strength of R of being a domain specific language has also become one of its weaknesses. Since new research findings in statistical computing are split up over several languages like R, Matlab or SciPy[1] it often becomes difficult to compare new methods with established ones. Since it is also hard to interface those languages due to different architectures, data storage mechanisms there is often no other way than to reimplement new methods in a different programming language to create a common scope.

An example for a well perceived new finding in statistical computing is the NMMSO-Algorithm by Jonathan E. Fieldsend [1]. It won the niching competition in 2015 held by the CEC and is only written in Matlab. Since the chair 'Information Systems and Statistics' at the Westfälische Wilhelms-Universität Münster, Germany is mainly concentrating its work on Statistical Computing in R an implementation of this algorithm became interesting.

As part of this Seminar Project in the context of the Seminar 'Statistical Computing in R' a reimplementation of the NMMSO algorithm in R will be presented. During this technical documentation, the general function of the algorithm and the used test cases by the CEC will be shown. Afterwards the structure and used techniques and libraries, as well as problems and pitfalls due to the different behaviours of R and Matlab, will be shown. The documentation will be closed by the benchmarking results and different test cases.

It was the goal of this project to keep up high comparability with the original code, to ensure the correct functionality and easily implement changes to the original codebase in this program.

. . . write a bit more here.

---

2 The Algorithm

2.1 General Function

The starting point of the project was the paper provided by Dr. Jonathen E. Fieldsend [1] on the Niching Migratory Multi-Swarm Optimiser (NMMSO) algorithm. NMMSO is a multi-modal optimiser which relies heavily on multiple swarms which are generated on the landscape of an function in order to find the global optimum. It is build around three main pillars: (1) dynamic in the numbers of dimensions, (2) self-adaptive without any special preparation and (3) exploitative local search to quickly find peak estimates [1].

Multi-modal optimisation in general is not that different from well known and widely discussed single-objective optimisation, but in difference to it the goal of the algorithms in the multi-modal is not to find just one single optimising point but all possible points [1]. In order

[1] SciPy is a common library for the Python Programming language which brings Statistical Computing capabilities to the language.

to do so, many early multi-modal optimisation algorithms needed highly defined parameters [TODO: quote needed].

maybe it would be interesting to write a few more lines about the history of evolutionary algorithms here?

Newer algorithms fall in the field of self-tuning and try to use different mathematical paradigms like nearest-best clustering with covariance matrices [2] and strategies like storing the so far best found global optima estimators to provide them as parameters for new optimisation runs [3]. Contradictory to that NMMSO goes another way and uses the the swarm strategy in order to find which store their current [1]

In order to do so NMMSO follow a strict structure which can be seen in the following pseudo-code

```
nmmso(max_evals, tol, n, max_inc, c_1, c_2, chi, w)
    S: initialise_swarm(1)
    evaluations := 1
    while evaluations < max_evals:
        while flagged_swarms(S) == true:
            {S, m} := attempt_merge(S, n, tol)
            evals := evals + m
        S := increment(S, n, max_inc, c_1, c_2, chi, w)
        evals := evals + min(|S|, max_inc)
        {S, k} := attempt_separation(S, tol)
        evals := evals + k
        S := add_new_swarm(S)
        evals := evals + 1
    {X*, Y*} := extract_gebsest(S)
    return X*,Y*
```

This structure wasn't modified during the reimplementation of NMMSO to keep comparability and the possibility to fix bugs at a high level. The only newly introduced setting was the possibility to modify the c_1, c_2, chi, w as parameters from the outside. In the original version those parameters are part of the program code.

What else about the algorithm need to be explained that isn't explicitly part of the implementation?

2.2 CEC

The IEEE Congress of Evolutionary Computation (CEC) is one of the largest, most important and recognised conferences within Evolutionary Computation (EC). It is organised by the IEEE Computational Intelligence Society in cooperation with the Evolutionary Programming Society, and covers most of the subtopics of the EC.

In order to validate the potential of the NMMSO algorithm, it was submitted to the IEEE CEC 2013 held in Cancun, Mexico. Here, Dr. Fieldsend was provided with some multimodal benchmark test functions with different dimension sizes and characteristics, for evaluating

niching algorithms developed by Dr. Xiaodong Li, Dr. Andries Engelbrecht and Dr. Michael G. Epitropakis [3]. They state that even if several niching methods have been around for many years, further advances in this area have been hindered by several obstacles; most of the studies focus on very low dimensional multimodal problems (2 or 3 dimensions) making this more complicated to asses theses methods' scalability to high dimensions with better performance. The benchmark tool includes 20 test functions (in some cases the same function but with different dimension sizes), which includes 10 simple, well-known and widely used benchmark functions, based on recent studies, and more complex functions following the paradigm of composition functions. In the following section, they will be briefly explained:

- `F1: Five-Uneven-Peak Trap (1D)`
- `F2: Equal Maxima (1D)`
- `F3: Uneven Decreasing Maxima (1D)`
- `F4: Himmelblau (2D)`
- `F5: Six-Hump Camel Back (2D)`
- `F6: Shubert (2D, 3D)`
- `F7: Vincent (2D, 3D)`
- `F8: Modified Rastrigin - All Global Optima (2D)`
- `F9: Composition Function 1 (2D)`
- `F10: Composition Function 2 (2D)`
- `F11: Composition Function 3 (2D, 3D, 5D, 10D)`
- `F12: Composition Function 4 (3D, 5D, 10D, 20D)`

All of the test functions are formulated as maximisation problems. F1, F2 and F3 are simple 1D multimodal functions, while F4 and F5 are simple 2D functions and not scalable. F6 to F8 are scalable multimodal functions. The number of global optima for F6 and F7 are determined by the dimension. However, for F8, the number of global optima is independent from the dimension, therefore it can be controlled by the user. F9 to F12 are scalable multimodal functions constructed by several basic functions with different properties (Sphere function, Grienwank, Rastrigin, Weierstrass and the Expanded Griewank's plus Rosenbrock's function). F9 and F10 are separable, and non-symmetric, while F11 and F12 are non-separable, non-symmetric complex multimodal functions. The number of global optima in all of the composition functions is independent from the number of dimensions, therefore can be controlled by the user [3].

Maybe write each math equation or the R code

https://en.wikipedia.org/wiki/IEEE_Congress_on_Evolutionary_Computation

---

3 The Implementation

3.1 Structure of the project

After analysing the algorithm provided in Matlab by Dr. Fieldsend, it was decided to first translate each of the functions into the R programming language. At first instance, this task

seemed to be simple because most of the functions were basically managing matrices and vectors, but later this became a problem that will be addressed in the pitfalls' section of this paper.

Once all the NMMSO functions existed in R and having the input data, the testing phase started. It has be said, that one of the biggest problems when you code an already existing program into another programming language, is the different behaviours corresponding to each object (in case of an object-oriented language) or its main structure. The first runs came with several errors regarding the matrix generation and handling, slowing down the project in a near future. Using GitHub, it was easier to attack these problems in parallel, having one developer reviewing different functions and the other one, fixing other bugs and continue the testing phase. Also, this was achieved in an easier way, thanks that each function was coded in an independent R file, making easier and faster the debugging and the fixing of each problem.

During the developing time, an issue raised with the CEC benchmark tool. In order to compare the R implementation of the NMMSO algorithm with the original one, it was mandatory to use this tool to test each of its functions with the new algorithm and compare results. After several complications with the original test suite (these complications will be addressed in the pitfalls' section), it was decided to recode each of the functions as an independent R package to avoid any further complication and having an easier and more trustworthy comparison of the NMMSO algorithm in R.

3.2 Pitfalls and Problems

test

3.3 Benchmark and Comparison

To compare the nmmsoR with the original NMMSO the CEC test cases were used to run the same benchmarks as in the original submission [1]. There 4 different Ratios were used to measure the performance of certain algorithms. Three of those measures (Peak Ratio, Success Ratio and Convergence Speed) have been introduced in [3] to create a common point of comparison. The fourth ratio is special for the nmmso algorithm since it tracks the number of swarms over the iterations of the algorithm. Nmmso.R uses the same measures to reach the highest comparability possible.

Table 1: Success Ratio over given runs

	0.1	0.01	0.001	0.0001	0.00001	runs
F1	1	1	1	1	1	17
F2	1	1	1	1	1	15
F3	1	1	1	1	1	17
F4	1	1	1	1	1	17
F5	1	1	1	1	1	14
F6	1	1	1	1	0	14

	0.1	0.01	0.001	0.0001	0.00001	runs
F7	1	1	1	1	1	14
F8	1	1	1	1	0.8333	6
F9	1	1	1	1	1	8
F10	1	1	1	1	1	14
F11	1	1	1	1	1	14
F12	1	1	1	1	1	14
F13	1	1	1	1	1	14
F14	1	1	1	1	1	14
F15	1	1	1	1	1	7
F16	0	0	0	0	0	2
F17	0.5	0	0	0	0	2
F18	0	0	0	0	0	1
F19	0	0	0	0	0	2
F20	0	0	0	0	0	1

Table 2: Convergence Rates over given runs

	0.1	0.01	0.001	0.0001	0.00001	runs
F1	613.7	783.6	1044	1220	1485	17
F2	156.4	267.8	408.4	580.3	690.7	15
F3	47.65	181.4	262.1	361.6	490.8	17
F4	489.8	745.4	974.1	1131	1447	17
F5	92.07	251.9	383.6	583.4	830.5	14
F6	18356	23594	29162	41493	2e+05	14
F7	7636	8249	9488	10528	12298	14
F8	177212	203759	233870	266790	320498	6
F9	180477	185068	204954	216185	230947	8
F10	858.7	1324	1699	2226	2793	14
F11	3773	5175	6774	7966	8471	14
F12	16695	22960	35280	40981	46842	14
F13	8931	12949	17334	23123	25213	14
F14	35471	41272	62197	70296	82930	14
F15	72129	82885	136540	158129	164531	7
F16	4e+05	4e+05	4e+05	4e+05	4e+05	2
F17	268887	4e+05	4e+05	4e+05	4e+05	2
F18	4e+05	4e+05	4e+05	4e+05	4e+05	1
F19	4e+05	4e+05	4e+05	4e+05	4e+05	2
F20	4e+05	4e+05	4e+05	4e+05	4e+05	1

Table 3: Mean Peak Ratio over given runs

	0.1	0.01	0.001	0.0001	0.00001	runs
F1	1	1	1	1	1	17
F2	1	1	1	1	1	15
F3	1	1	1	1	1	17
F4	1	1	1	1	1	17
F5	1	1	1	1	1	14
F6	1	1	1	1	0	14
F7	1	1	1	1	1	14
F8	1	1	1	1	0.9938	6
F9	1	1	1	1	1	8
F10	1	1	1	1	1	14
F11	1	1	1	1	1	14
F12	1	1	1	1	1	14
F13	1	1	1	1	1	14
F14	1	1	1	1	1	14
F15	1	1	1	1	1	7
F16	0.08333	0	0	0	0	2
F17	0.875	0.8125	0.8125	0.8125	0.6875	2
F18	0.5	0.5	0.5	0.5	0.5	1
F19	0.4167	0.4167	0.3333	0.3333	0.3333	2
F20	0.25	0.25	0.25	0.25	0.25	1

3.4 Testing and alternative parameter settings

test

4 Discussion

test

5 Conclusion

test

6 Acknowledgements

Thanks to everybody!

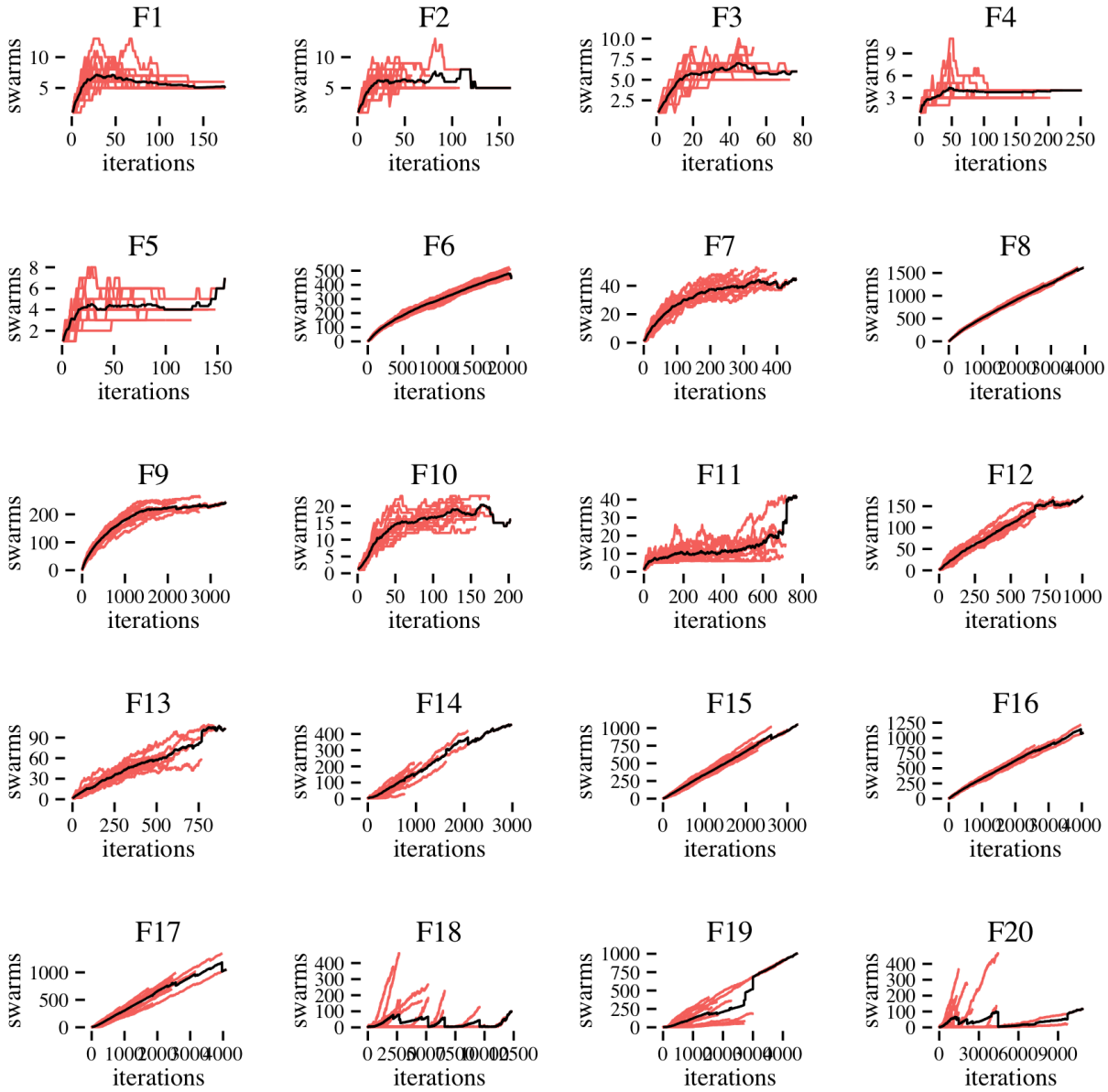

Figure 1: plot of chunk trend curve of kept swarms over all 20 functions.

[1] J.E. Fieldsend, Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser, in: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, 2014: pp. 2593–2600.

[2] M. Preuss, Niching the cMA-eS via nearest-better clustering, in: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, ACM, New York, NY, USA, 2010: pp. 1711–1718. doi:10.1145/1830761.1830793.

[3] M.G. Epitropakis, X. Li, E.K. Burke, A dynamic archive niching differential evolution algorithm for multimodal optimization, in: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, 2013: pp. 79–86.