Heuristic Optimization and Dynamical System Safety Verification

Todd W. Neller*
Knowledge Systems Laboratory, Stanford University

August 25, 1997

Abstract

Given a heuristic estimate of the relative safety of a hybrid dynamical system trajectory, we transform the initial safety problem for dynamical systems into a global optimization problem. We compare untuned performance of several Simulated Annealing and Multi Level Single Linkage method variants, and discuss the dynamic use of knowledge gained during optimization.

1 Introduction

Given a simulated hybrid dynamical system S, a set of possible initial states I, and a set of "unsafe" states U, we wish to verify nonexistence of an S-trajectory from I to U within t_{max} time units. We call this the *initial safety problem.* Suppose we are given an approximate measure of the relative safety of a trajectory. More specifically, let f be a function taking an initial state i as input, and evaluating the S trajectory from i such that $f(i) = 0$ if and only if the S-trajectory from i enters U within t_{max} time units, and $f(i) > 0$ otherwise. Then verification of the initial safety problem can be transformed into the global optimization (GO) problem:

$$\min_{i \in I}(f(i)) \overset{?}{>} 0$$

GO methods may therefore terminate when i is found such that $f(i) = 0$. Given that f does not generally have an analytic form, we do not assume the availability of derivatives. Since each evaluation of f may require a computationally expensive simulation, we are particularly interested in GO methods which perform relatively few evaluations of f. In this context, we compare several variants of Simulated Annealing (SA) and Multi Level Single Linkage (MLSL) methods and assess their suitability for our purposes. We discuss the use of knowledge of f gained in the course of GO, and consider the extent to which some GO methods assume special properties of the local optimization (LO) procedures they use.

*This work was supported by the Defense Advanced Research Projects Agency and the National Institute of Standards and Technology under Cooperative Agreement 70NANB6H0075, "Model-Based Support of Distributed Collaborative Design". Author's address: Knowledge Systems Laboratory, Gates Building 2A, Stanford University, Stanford CA 94305-9020. `neller@ksl.stanford.edu`

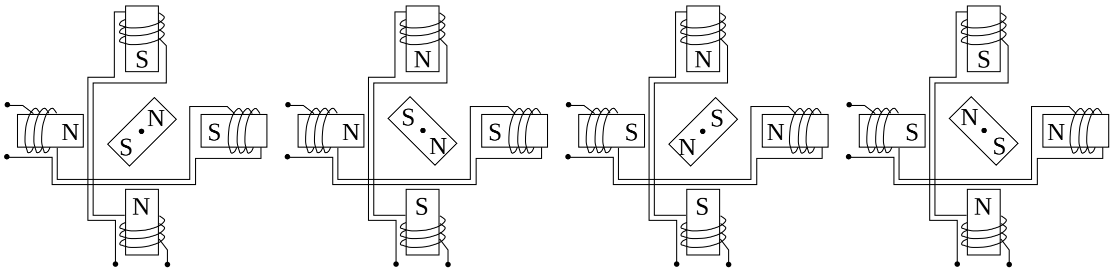


Figure 1: Simple Stepper Motor Stepping

2 Motivation

Our research was largely motivated by the following safety verification task: Given bounds on the system parameters of a stepper motor (e.g. viscous friction, inertial load), bounds on initial conditions (e.g. angular displacement and velocity), and an open-loop motor acceleration control, verify that no scenario exists in which the motor stalls. We model the motor's continuous dynamics using ODEs given in [9]:

$$\begin{aligned}
\dot{\theta} &= \omega \\
\dot{\omega} &= (-i_a N_b \sin(N\theta) + i_b N_b \cos(N\theta) - D\sin(4N\theta) - F_v\omega - F_c \mathrm{sign}(\omega) - F_g)/(J_l + J_m) \\
\dot{i_a} &= (V_a - i_a R + \omega N_b \sin(N\theta))/L \\
\dot{i_b} &= (V_b - i_b R - \omega N_b \cos(N\theta))/L
\end{aligned}$$

where θ and ω are motor shaft angular displacement and velocity, i_a and i_b are coil A and B current, V_a and V_b are coil A and B voltage, R and L are coil resistance and inductance, N is the number of rotor teeth, N_b is the maximum motor torque per amp, D is the maximum detent torque, F_v is the viscous friction, F_c is the Coulomb friction, F_g is the gravitational torque load, and J_l and J_m are load and motor shaft inertia. For this system we classify a stall as deviation of $\frac{\pi}{N}$ or more radians from the current desired θ equilibrium.

The motor is stepped by reversing polarity of the coil voltages in alternation (see Figure 1). Changes to coil voltages occur on such a small time scale that their continuous simulation is judged unnecessary for modeling dynamics relevant to the verification task. Voltage changes were therefore approximated as discrete events. Our acceleration control is open-loop: At fixed intervals the motor is stepped according to an acceleration table. We can express such a system as a nonlinear hybrid automaton as shown in Figure 2.

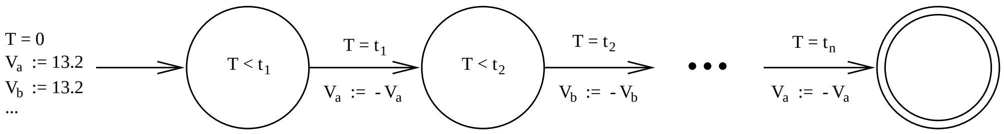


Figure 2: Stepper Motor Nonlinear Hybrid Automaton

First, we note that there is no apparent "geometrically linear hybrid system"[1] approximation with which we could apply the tools of computational geometry, but simulation is feasible.

[1] i.e. restricted to constant first derivatives; "geometrically" as opposed to "algebraically"

Next, we note that our verification is concerned with a fixed initial time interval (i.e. during acceleration) and is therefore an initial safety problem. Finally, we note that we can compute minimum angular displacement from a stall state over all simulation states as a simple heuristic to numerically rate the relative safety of safe trajectories. We can now ask, "For all possible system parameters and initial states, are all simulation trajectories rated safe?" Put another way, "Is the minimum heuristic evaluation of all possible simulations greater than zero?" If we can answer this optimization question positively, we have verified safety of our hybrid system.

One could argue that such optimization is *not* verification, that one cannot exhaustively simulate all possibilities and can therefore have no guarantees. One can only use such optimization for refutation. To this, we offer two responses: First, if one has additional knowledge of chacteristics of one's heuristic evaluation function (e.g. Lipschitz conditions), then an intelligent optimization approach can utilize such characteristics to guarantee a strictly positive minimum with enough testing. The key is to provide a heuristic evaluation that induces a helpful search landscape without itself become overly burdensome computationally. Second, if one has no such knowledge about the heuristic, the absence of verification techniques well-suited to non-trivial dynamics leaves good global optimization as our best assurance. Our desire is to develop an information-based GO method which, when halted without finding an unsafe trajectory, provides some measure of the thoroughness of its search.

This said, we have endeavored to study a number of representative global optimization techniques in order to assess their suitability to our purpose and point the way towards future innovation.

3 Algorithms and test problems

In this section, we describe the global optimization (GO) algorithms used in this study, the local optimization procedures used by them, and the test functions to be minimized. Author-supplied default settings were used for GO algorithms when possible. Otherwise, reasonable parameters were held constant throughout testing. Since our goal is to perform a computationally expensive optimization, we would desire an algorithm which reliably and efficiently gives the desired result without tuning. Experienced users of such algorithms applying problem- and domain-specific knowledge to the choice of options and parameters could expect to yield better results.

The first set of algorithms we consider are variants of simulated annealing (SA) [10, 8]. SA algorithms are theoretically guaranteed to find the global minimum of a function provided that the annealing schedule starts with sufficiently high temperature and cools sufficiently slowly. However, this guarantee comes at great expense in terms of function evaluations. Finding a suitable annealing schedule which balances the tradeoff of reliability versus efficiency is key to the practicality of SA for our purposes.

`AMEBSA` [11, pp.451-455] performs SA by modifying a downhill simplex method [11, pp.408-412] such that actual function values of simplex points and possible replacement points are perturbed according to the temperature parameter when making move decisions. Since `AMEBSA` has no default annealing schedule, we have chosen to use the one supplied in the authors' example [12, pp.182-184]. `ASA`[2] [7], "adaptive simulated annealing", is a SA variant that relies on randomly importance-sampling the search space and adapts separate annealing schedules for each parameter. The automatic adaptation of the annealing schedule trades off reliability for efficiency. `SALO` [4] seeks to combine the theoretical guarantees of SA with the efficiency of local optimization (LO). `SALO` on f is SA on f', where f' is f transformed by LO. At each point that

[2] `ASA` software developed by Lester Ingber and other contributors is available at URL `http://www.ingber.com/` or `ftp://ftp.ingber.com`.

SA evaluates, LO takes place and the value of the local minimum is returned. This is intended to "flatten" f and speed convergence to the global minimum. In both implementations described here and in [4], `ASA` is used as the SA method. In doing so, we again tradeoff reliability for efficiency. When each of these SA methods halts unsuccessfully, it is restarted from the lowest point found thus far.

The second set of algorithms we consider are variants of Multi Level Single Linkage (MLSL) [14]. MLSL uniformly, iteratively samples the search space and performs LO selectively. For each iteration, a new batch of points is evaluated. For each point sampled, LO takes place if there exists no lower sampled point within a critical distance.[3]. `MLSL1` is the original algorithm [14]. `MLSLD` is a variant of `MLSL1` which assumes that the LO procedure is deterministic and should therefore never be repeated from the same sampled point. `MLSLO` orders optimizations for each iteration by ascending function value of sampled points. `MLSLOD` has both variations. `MLSLSA` alternates iterations of `MLSLOD` with runs of `ASA`, using the current minimum as the initial point for `ASA`. `LMLSL` is a variant of `MLSL1` which performs "lazy" function evaluation. That is, the function value of a point is only evaluated when it becomes necessary. This avoids the relatively large initial cost when optimizing simple functions. $\texttt{LMLSL}_\epsilon$ is `LMLSL` using an ϵ-descent LO procedure. An epsilon-descent procedure guarantees that, for a step greater than ϵ, the function values at epsilon intervals are sequentially descending.

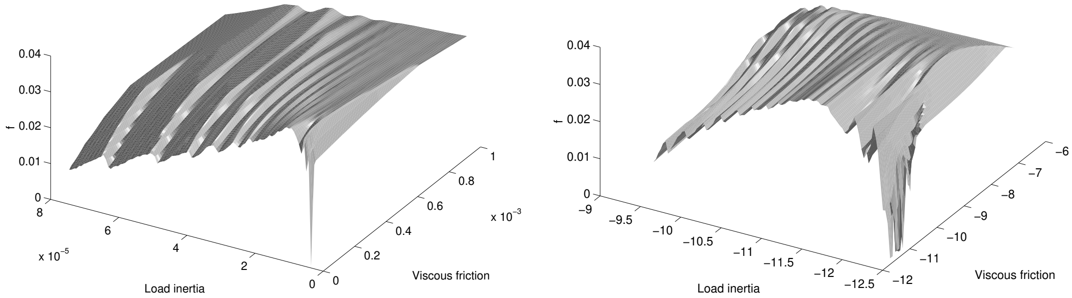
`RANDLO` simply performs random local optimizations and is intended to provide a baseline for understanding how well LO knowledge is used by `SALO` and MLSL methods. `MONTE` is a Monte Carlo method.

We next describe the local optimization procedures used by some of these global optimization algorithms. `FMINU` and `CONSTR` are MATLAB™ optimization functions [5]. `FMINU` performs unconstrained optimization using a quasi-Newton method with a BFGS formula for updating the Hessian matrix approximation. $\texttt{FMINU}_\epsilon$ is an ϵ-descent modification of `FMINU`. `CONSTR` performs constrained optimization using a sequential quadratic programming method. We supply search space bounds and no additional constraints. `YURETMIN` is a variant of Yuret's Masters thesis Procedure 4-1 [15, p.33] which allows specification of search space bounds.

Finally, we reference the objective functions used for comparing the global optimization algorithms. The first part of our study uses functions selected from GO literature and algorithm demonstrations in order to reveal their relative merits. `RAST` is a scaled Rastrigin function [4]. `HUMP` is the six-hump camelback function [2]. `G-P` is the Goldstein-Price function [2]. `GW1` and `GW100` are 6-dimensional Griewank functions with bounds of each dimension $[-1, 1]$ and $[-100, 100]$ respectively [4]. `SWISS` is a 4-D paraboloid with a lattice of many circular pits [12]. `CMMR` is a 4-D paraboloid with a grid of deep troughs [3]. `GW100`, `SWISS`, and `CMMR` have many local minima. `RAST` has a moderate number. `HUMP`, `G-P`, and `GW1` have few. `RAST`, `GW100`, `SWISS`, and `CMMR` are generally paraboloid in shape with different local minima "traps". All slope up to the bounds of the search space.

The second part of our study concerns the motivating example for this research. Test function `STEP1` takes as input two parameters (viscous friction and load inertia) of the stepper motor model, simulates acceleration of the motor, and performs a simple heuristic evaluation of the trajectory by computing the minimum distance to a stall state (0 if stalled). Such a heuristic function is often simple to construct. `STEP2` is `STEP1` logarithmically scaled so as to focus on the unsafe region of the parameter space. These functions are shown in Figure 3.

[3]We used the critical distance parameter $\zeta = 2$ with 100 points generated per iteration.



(a) `STEP1` (b) `STEP2`

Figure 3: Stepper Motor Test Functions

4 Results

Our first tests made use of LO procedure `FMINU` where applicable. 100 optimization trials were performed for each objective function with a maximum of 10000 function evaluations permitted per trial. Each objective function was offset (if necessary) to have a global minimum value of 0. A successful trial was one in which the optimization routine found a point with function value less than .001 within 10000 function evaluations.

Each entry in the table of results (Figure 4) shows the number of successful trials (upper left) and the average number of function evaluations for such trials (lower right).

	`RAST`	`HUMP`	`G-P`	`GW1`	`GW100`	`SWISS`	`CMMR`
`AMEBSA`	16 39	**100** **40**	90 222	100 86	0 N/A	100 1340	2 5674
`ASA`	**100** **404**	100 225	100 1042	100 197	2 6003	100 903	**100** **3756**
`SALO`	100 585	100 65	100 97	**100** **85**	**95** **4501**	100 163	0 N/A
`MLSL1`	100 872	100 154	100 170	100 185	47 4315	100 239	0 N/A
`MLSLD`	100 636	100 154	100 170	100 185	60 4492	100 238	0 N/A
`MLSLOD`	100 556	100 130	100 132	100 173	52 4370	100 253	0 N/A
`MLSLSA`	100 544	100 131	100 130	100 174	22 2609	100 254	99 5019
`LMLSL`	100 847	100 105	100 118	100 96	50 4508	100 187	0 N/A
`LMLSL`$_\epsilon$	100 638	100 96	100 109	100 93	53 3864	100 192	0 N/A
`RANDLO`	100 706	100 70	**100** **96**	**100** **85**	58 4008	**100** **146**	0 N/A

Figure 4: Successful global optimization trials and average function evaluations

Giving the best performance in nearly half of the tests, `RANDLO` performed surprisingly well, especially for `SWISS` which has a 4-D lattice of numerous "traps". As `RANDLO`'s LO procedure, `FMINU` is clearly rarely caught in such traps. Since both trap and non-trap regions are paraboloid surfaces, they effectively "point" to the global minimum for LO procedures such as `FMINU`. The simple but important observation here is that local optimization does not necessarily find the *nearest* local optimum. We next observe that both `SALO` and MLSL each rely somewhat on *nearness* of LO, and will later turn our attention to the relationship between the global and local layers of each. `FMINU`, which assumes f is continuous, behaved understandably poorly for highly discontinuous `CMMR`. Thus all methods dependent entirely on LO failed all `CMMR` trials. Given that the characteristics of f may not be well understood, this means that a less efficient LO procedure making fewer assumptions would likely be better suited to our purposes.

`SALO` yielded performance similar to that of `RANDLO` where few LOs sufficed and significantly better where more local optima trapped LO (e.g. in `RAST` and `GW100`). At the heart of `SALO`'s design is the following intention: "SA helps in locating good regions of the search space, while the local optimizer is used to rapidly hit the optimum." [4] It is clear from this comparison that `SALO` does indeed successfully apply SA on f' to find good regions of f. When comparing `ASA` with `SALO`, it also appears that the cost for transforming $f \overset{\texttt{FMINU}}{\rightarrow} f'$ is usually more than compensated by the efficiency gained.

`SALO` was designed with hope that f' would be a "simpler" surface than f, reflecting the function value of the nearest optimum. Interestingly, the designers' experiments utilized Yuret's LO procedure which has short term memory and takes increasingly greater steps downhill as success allows. Such a LO procedure can possibly pass over nearest local minima as step size becomes large. Also Yuret's procedure, being stochastic, doesn't simply transform one surface to another. Nevertheless, their experiments and ours indicate that `ASA` is able to handle such LO output gracefully in the long run. The fact that `SALO` outperforms `RANDLO` for harder optimization problems is specifically a property of SA and more generally a form of learning. One can view the changing state probability distribution of SA as a gradual accumulation of knowledge about the location of the global minimum. While such learning is effective given a suitable annealing schedule, it is also weak. Heavily traversed local minima may be heavily traversed again. All but one of the function evaluations made in LO are ignored. Much information is wasted. Nonetheless, `SALO`'s performance was impressive.

Performance of MLSL methods, though similar to that of `RANDLO`, yields little to commend them over `RANDLO`. That selective uniform random LO should perform worse than unselective uniform random LO suggests an assumption in MLSL which is not met in our study. Following the analysis more closely in both [13] and [14], we see that MLSL's LO procedure is assumed to be an *ϵ-descent procedure* such that the current critical distance effectively bounds the step size of LO.[4] We therefore modified `FMINU` to be an ϵ-descent procedure and tested $\texttt{LMLSL}_\epsilon$ for comparison. Although $\texttt{LMLSL}_\epsilon$ is somewhat of an improvement over `LMLSL`, it is still generally worse than `RANDLO`. ϵ-descent does not therefore appear to help us much. We conjecture that MLSL methods dominate `RANDLO` for objective functions where LO is trapped in many minima, and that `SALO` dominates MLSL methods for such objective functions in our study because our f'-surfaces are easily globally optimized with LO. To elucidate the latter point, consider `RAST`, `GW100`, and `SWISS`. LO roughly transforms each into a paraboloid of plateaus. LO of such LO-transformed functions can then efficiently lead to the global optimum. We can view the task of global optimization as *recursive local optimization.* The base-level LO_0 takes advantage of whatever information about f is available (continuity, gradients, etc.), the next level LO_1 is suited to the class of one's LO_0-transformed function f', and so on. We may stop after arbitrarily

[4] This is nowhere mentioned in survey [1] and is not emphasized elsewhere in the literature.

	STEP1	STEP2
ASA	0 N/A	2 497
SALO	10 80	5 202
MLSLOD	10 127	10 191
LMLSL	10 163	**10** **137**
RANDLO	**10** **78**	10 359
MONTE	0 N/A	6 469

(a) CONSTR

	STEP1	STEP2
ASA	0 N/A	2 497
SALO	7 387	9 198
MLSLO	4 790	10 231
LMLSL	3 389	**10** **169**
RANDLO	**9** **501**	10 172
MONTE	0 N/A	6 469

(b) YURETMIN

Figure 5: Results for STEP1 and STEP2

many (probably 2-3) LO levels and simply perform random local optimization. The role of each LO level is to enlarge the regions leading to global optima. We are currently developing such recursive local optimization methods.

Regarding MLSL methods, let us also note that, like SALO, they all but ignore information gained through LO. Uniformly sampled points are locally optimized based only on the values of sampled points within a critical distance. Again we find great waste of information gained at great expense.

AMEBSA gave mixed results which can likely be attributed to the lack of annealing schedule tuning. Perhaps an adaptive annealing schedule would make AMEBSA more suitable for such problems. ASA's efficiency was unpredictable, although it was perhaps the most reliable method for this set of objective functions.

While these functions may give a general indication of the relative strengths of these methods without tuning, the functions share a common property undesirable for our purposes: The *unconstrained* global minimum is never located at or beyond the bounds of the search space. Therefore, our optimization methods need not perform well along the bounds of our search space. It is for this reason that unconstrained FMINU was suitable for use with such global optimizations. We used this as an opportunity to try two constrained LO procedures CONSTR and YURETMIN for the stepper motor test problems STEP1 and STEP2. For this testing, we performed 10 trials to find a function value of 0 with a maximum of 1000 function evaluations per trial. The results appear in the tables of figure 5.

Since both STEP1 and STEP2 have a small number of local minima along the bounds of the search space, behavior of LO again figured most significantly in our results. Despite the fact that much of the search space slopes downward away from the corner where failures occur, CONSTR had a bias towards looking in that particular corner. It was thought that STEP2 (log-log scaled STEP1) would be an easier function to optimize, but this was not the case. Not only was the global minimum basin expanded, but nearby local minima also expanded, trapping LO more often.

ASA's function evaluation expenses were such that it was outperformed by MONTE. The remaining LO-based methods performed similarly overall. The cost of computing simple heuristic information about relative safety of trajectories is usually more than compensated by efficiency in discovering unsafe trajectories through optimization. For both LO procedures, RANDLO gave best performance for STEP1 and LMLSL gave best performance for STEP2. Although there was no universal "winner" among global optimization procedures, it is encouraging to note that procedures such as SALO and LMLSL could be run in parallel to achieve respectable, more reliable results. The choice of LO procedure proved very significant for performance, which again underscores the importance of developing robust, efficient LO procedures suited to large classes of functions.

5 Conclusions

A powerful approach to initial safety verification is to transform the problem into an optimization problem and leverage the power of efficient optimization methods. This is accomplished by

- providing a good heuristic evaluation function f,
- choosing an efficient local optimization procedure well suited to f, and
- applying a global optimization procedure such as SALO with ASA or RANDLO for which one's local optimization procedure is well suited.

While no global optimization procedure was generally dominant, random local optimization seemed best suited for objective functions with few minima, and SALO with ASA seemed best suited for objective functions with many minima. By making use of ASA for SA, one both avoids the need to specify an annealing schedule and benefits from its relative efficiency among SA algorithms. Although one is encouraged to make use of ASA's options to improve performance, we have not done so and have been pleased with most results nevertheless.

SALO and MLSL methods perform global optimization with global and local search phases, and rely on local optimization for efficiency. However, both methods make little or no use of information gained in the course of local optimization. We believe that great progress will be made in global optimization when global optimization and local optimization are seamlessly integrated to share knowledge gained of f. Where evaluation of f is computationally expensive, it is worth computational expense to utilize such knowledge for the efficiency of global optimization. To this end, we are currently developing a set of information-based optimization techniques where each optimization step is chosen with respect to the information gained thus far.

References

[1] C. Guus E. Boender and H. Edwin Romeijn. Stochastic methods. In Horst and Pardalos [6], pages 829–869.

[2] F.H. Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *I.B.M. J. R&D.*, Sept 1972.

[3] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Trans. Mathl. Software*, 13(3):262–279, 1987.

[4] Rutvik Desai and Rajendra Patil. SALO: combining simulated annealing and local optimization for efficient global optimization. In J.H. Stewman, editor, *Proceedings of the 9th Florida AI Research Symposium (FLAIRS-'96)*, pages 233–237, St. Petersburg, FL, USA, 1996. Eckerd Coll.

[5] Andrew Grace. *Optimization Toolbox.* The Mathworks Inc., 24 Prime Park Way, Natick, MA 01760-1500 USA.

[6] Reiner Horst and Ranos M. Pardalos, editors. *Handbook of Global Optimization.* Kluwer Academic, Dordrecht, Netherlands, 1995.

[7] Lester Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54, 1996.

[8] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[9] Albert C. Leenhouts. *Step Motor System Design Handbook.* Litchfield Engineering, Kingman, Arizona, USA, 1991.

[10] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the art of scientific computing - 2nd Ed.* Cambridge University Press, Cambridge, 1992.

[12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes Example Book (C) - 2nd Ed.* Cambridge University Press, Cambridge, 1992.

[13] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods; part I: multi level methods. *Mathematical Programming*, 39:27–56, 1987.

[14] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods; part II: clustering methods. *Mathematical Programming*, 39:57–78, 1987.

[15] Dinez Yuret. From genetic algorithms to efficient optimization. Master's thesis, Massachusetts Institute of Technology, May 1994.