Department of Physics and Astronomy
University of Heidelberg

Bachelor Thesis in Physics
submitted by

Maximilian Argus

born in Hamburg, Germany

1990

Electric Field Optimization of a Rydberg Atom Experiment

This Bachelor Thesis has been carried out by Maximilian Argus at the
Physikalisches Institute in Heidelberg
under the supervision of
Prof. Dr. Matthias Weidemüller

Abstract

Modern experiments with ultracold Rydberg atoms with application to many body physics and quantum information science, demand a high level of experimental sophistication to precisely control experimental parameters like external electric fields, as Rydberg atoms are very polarizable. In the experiment this is achieved by (a structure hosting) ¿10 individually controllable electrodes. However, the task of finding the optimal control voltages for these is complicated by incomplete knowledge of the charge distributions, including possible patch fields (making it particularly time consuming). To overcome this challenge we have applied evolutionary algorithms, a group of powerful search heuristics, to optimize the overall performance of our experiment. With particular focus on electric field control we asses the performance of several algorithms, on competing requirements of noise robustness and fast convergence, in solving two problems: cancellation of electric fields and and optimum guiding of field ionized Rydberg atoms to a MCP detector. Additionally Foreseeable applications to controlling quantum state evolution and engineering strongly correlated many body systems of interacting Rydberg atoms will be considered.

Introduction

Introduction introduction introduction

Contents

1 Concept **3**

2 Theoretical Background **4**

3 Optimization **5**
3.1 Mathematical Formulation 5
3.2 Background: Why look at multiple algorithms? 5
3.3 Evolutionary Algorithms 5
3.4 Genetic Algorithms 6
3.4.1 Generation 6
3.4.2 Evaluation 6
3.4.3 Selection 6
3.4.4 Variation 6
3.4.5 Exploration vs. Exploitation 7
3.5 Differential Evolution 7
3.6 Swarm Intelligence 8

4 Implementation **9**
4.1 Test Function 9
4.2 Comparison of Algorithms 9
4.2.1 Why choose DEA? 10

5 Appendix **14**
5.1 Test Problem Parameters 14
5.2 List of Optimization Algorithms Considered 14
5.3 Classification of Optimization Algorithms 15

Chapter 1

Concept

The aim of this thesis is the application of optimization methods to a Rydberg atom experiment. In the course of making one measurement several steps have to be completed sequentially. These start with collecting Rubidium atoms into a MOT, pre-cooling then and then loading them into a dipole trap where they are exited into Rydberg atoms. After this has been done the atoms can be imaged by a camera system. These steps form a cycle that is repeated every few seconds with variation of parameters in order to make measurements of the change in behavior of the Rydberg atoms dependends on the varied parameters. While many properties of the experimental setup are physically fixed, such as the position of the lasers, a relatively large number (77) that can be controlled by the software controlling the experiment, such as the timing of activating these lasers. The aim of this project is to add information feedback to this cycle, so that the experiment can choose new experimental parameters based the results of previous evaluations. The schematic for this is shown in figure 1.1

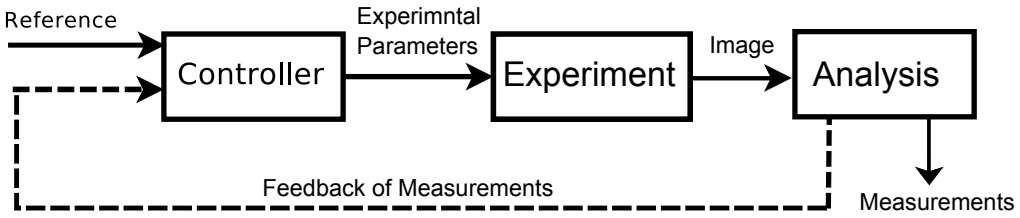


Figure 1.1: Information Feedback

Due to the discrete nature of experimental cycle evaluations and the large number of parameters as well as considerable measurement noise an optimization approach was chosen that generates and evaluates new experimental parameters. Using one can define the goal of the optimization as a function of several of the measurement results calculated by the analysis step.

Of particular interest is the control of experimental electrodes surrounding the Rydberg atoms. One of the experimental goal is creating minimal electric fields as the predictions that are made assume a field-free environment. Additionally it is important to generate a homogeneous filed as this prevents position dependent energies. When working with Rydberg atoms one mostly want to increase the principle quantum number n in order to scale with it the physical effects one is trying to measure, however the polarizability of Rydberg atoms scales with n^7 requiring fine control over the electric field if one wants to work with larger Rydberg atoms.

Chapter 2

Theoretical Background

Chapter 3

Optimization

3.1 Mathematical Formulation

A n dimensional real value optimization problem can be stated in the form

$$minf(\mathbf{x})$$

for

$$f : S \to \mathbb{R}$$

where

$$\mathbf{x} \in S \subseteq \mathbb{R}^n$$

A point $\mathbf{x}^* \in D$ is a global minimum if $f(\mathbf{x}^*) \leq f(\mathbf{x}^*) \forall \mathbf{x} \in S$
A point $\mathbf{x}^* \in D$ is a local minimum if in the surrounding $U \subseteq S$ of $\mathbf{x}^*$ it holds that $f(\mathbf{x}^*) \leq f(\mathbf{x}) \forall \mathbf{x} \in U$

3.2 Background: Why look at multiple algorithms?

There are a large number of optimization algorithms that have been developed in order to solve different problems. The key to applying optimization algorithms then is to find one suitable to solving kind of problems one is working with. Abstractly this is the converse to the No Free Lunch Theorem which states that the average performance of an optimization algorithms, when applied to the set of all optimization problems, does not outperform any other optimization algorithm or random walk. An algorithm is thus only, comparatively, suitable if it is able to utilize some form of problem specific information coupling the choice of algorithm to the problem at hand.

Due to the kind of problems to which optimization is to be applied to only the subset of Monte Carlo probabilistic global optimization algorithms will be considered. These algorithms sacrifice both evaluating the entire search space and the necessity of evaluating the function exactly in favor of a shorter run time. In these algorithms the choice of which candidates to evaluate is made by a heuristic which makes an induction based on previous evaluations. It is in this heuristic that is represents the problem specific information of the optimization algorithm.

3.3 Evolutionary Algorithms

Evolutionary Computation represents a subset of heuristic based approaches in which a set of possible solution candidates is maintained which the algorithm tries to refine over a number of generations. In the following sections we will introduce the different categories of algorithms used as well as the specific implementations of algorithms from these categories.

3.4 Genetic Algorithms

First we start with a simple genetic algorithm that will serve as a template for the algorithms to follow. In this algorithm we will encounter the common components of evolutionary algorithms, which are seen in the following block diagram.

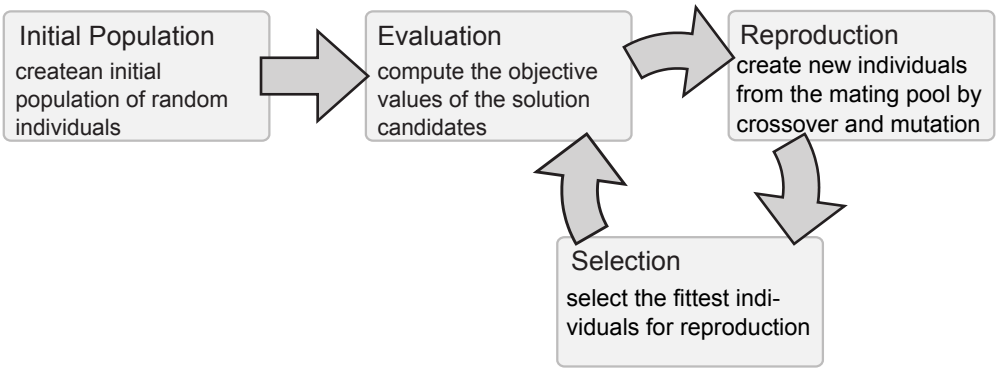


Figure 3.1: Block diagram of an Optimization Algorithm [1]

3.4.1 Generation

In order to start an optimization process one needs an initial population, as optimization is usually done over a restricted search space, one tries to choose an initial population of candidates one tries to sample the search space as even as possible. This decreases the chances of missing the location of the global minimum as well as avoids biasing the optimization to any particular region. This is usually achieved by using a uniform random distribution. An alternative is to use quasi-random sampling based on Halton sequences which are more equidistributed. This reduces the chances of evaluating candidates that are very similar at the cost of exploring the whole solution space.

3.4.2 Evaluation

Following the initial population choice these candidates are evaluated. In our case this is the most time consuming part of the optimization process as it requires an completion of the experimental cycle, thus there is not a lot that can be done to optimize this step.

3.4.3 Selection

Once the population has been evaluated the better candidates need to be selected as parents of the following generation. The aim here is to have the choice of parents be influenced by the fitness but not doing this too strongly as it would decrease the diversity of the population.

3.4.4 Variation

Given two different parents one can create a child candidate by mixing the parameter vectors of the two parents. This process is called a crossover operation, the most basic of which is the one point crossover where every value up to a certain index is copied from one parent, and the following entries are copied from the other parent. Another way of varying a child is through "mutation" that is performed by adding Gaussian noise to the element of a child vector.

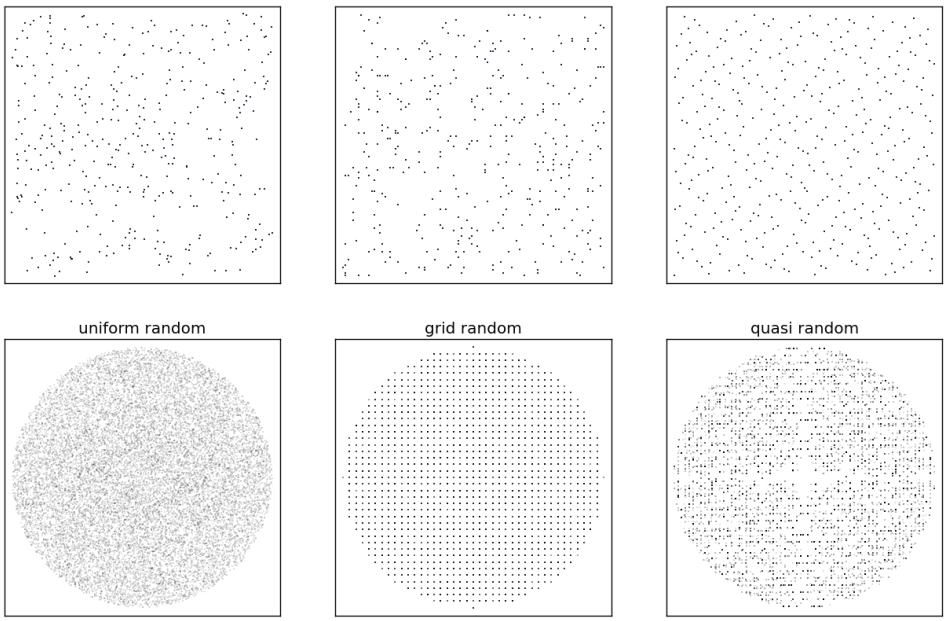


Figure 3.2: Three different sampling methods. The upper plot shows the similar looking positions sampled on a 1x1 square. The lower plot shows the relative distance of each point to all of its neighbors in a .2 radius. [2]

3.4.5 Exploration vs. Exploitation

One of the treadeoffs inherent in global optimization is the choice between exploration and exploitation. Exploration is the unbiased search for solution candidates in the whole search space whereas exploitation consists of trying to follow the lead of good solution candidates that have already been found. When the risk of an algorithm exploring too much is greatly increased run time it is more likely to find the global optimum conversely an algorithm that exploits good solution candidates too strongly will follow initial leads and get stuck in a local minimum. There are a number of ways that this behavior can be controlled in during the selection and variation stages of an algorithm. The population size chosen also has a strong effect.

3.5 Differential Evolution

Differential evolution is an optimization algorithm belonging to the Evolution Strategy family of optimization algorithms developed by Storn and Price [3]. It distinguishes itself by a special kind of three parent "differential" crossover variation that follows the form:

$$\mathbf{v}_i = \mathbf{x}_j + F \cdot (\mathbf{x}_k - \mathbf{x}_l)$$

Here $j, k, l \in \{1, 2, \ldots, NP\}/\{i\}$ are random vector indexes and $F \in [0, 2]$ is a weighting factor which controls the weight of the differential variation. However this new vector is not used directly, a binomial crossover with the initial vector $\mathbf{x_i}$ is performed, in which each element is chosen as the result of an independent binomial experiment.

$$\mathbf{x}'_{i,m} = \begin{cases} \mathbf{v}_{i,m} \text{ for } (r_m \leq CR) \vee (r_i = j) \\ \mathbf{x}_{i,m} \text{ for } (r_m > CR) \wedge (j \neq r_i) \end{cases}$$
$$\forall m \in \{1, 2, \ldots, D\}$$

Here r_i is a random integer index chosen once for each vector and $r_m \in [0, 1]$ in a random number chosen once for each vector element. This results in at least one component of the new vector coming from the varied one. The resulting vector $\mathbf{x}'$ is then kept if it has a better fitness values than its predecessor $\mathbf{x}$.

Differential evolution algorithms exist in a number of different variations that are named using the notation

$$DE/x/y/x$$

x represents the way in which the population should be varied. It can stand for "rand" or "best". With"rand" a randomly chosen vector is varied, as was previously explained and with "best" the best vector of the population is varied.

y is the number of difference vectors used, currently one.

z specifies the crossover scheme used, this can be "bin" for binomial crossover or "exp" for exponential crossover.

Using this notation the algorithm we have described would be represented by

$$DE/rand/1/bin$$

3.6 Swarm Intelligence

Particle swarm optimization was initially developed by Kennedy and Eberhart to model the behavior of a flock of birds. It was then noticed that the model could could be applied to do black-box optimization. In this method each particle has a position $\mathbf{x}$ and a velocity $\mathbf{v}$. In each iteration of the optimization the velocity is computed by the formula:

$$\mathbf{v} = \omega\mathbf{v} + \phi_p r_p(\mathbf{p} - \mathbf{x}) + \phi_g r_g(\mathbf{g} - \mathbf{x})$$

Where the parameter $\omega \in \mathbb{R}$ is the inertia weight. $\mathbf{p}$ and $\mathbf{g}$ are the best positions discovered by the particle and the swarm respectively. The attraction to these positions is controlled using the parameters ϕ_p, ϕ_g and varied stochastically using $r_p, r_g = U(O, 1)$. With each generation the velocity is applied to the position

$$\mathbf{x}' = \mathbf{x} + \mathbf{x}\mathbf{v}$$

This results in the swarm ideally converging on the global optimum position.

Chapter 4

Implementation

The application of optimization directly to the Rydberg experiment posed a number of difficulties in that the parameters for optimization on the physical experiment were very different from those which numerical optimization is usually applied to, while it is possible to evaluate a numerical function or simulation very quickly and or in parallel there was a strong constraint on the number of evaluations of solution candidates. In order to still be able to achieve convergence of the optimization algorithm this required a reduced dimensionality of the problem that we were attempting to solve. In addition to this physical measurements are also affected by noise. These conditions were used to choose an appropriate algorithm that could be applied the the experiment.

4.1 Test Function

In order to evaluate the different algorithms that are available a mathematical test problem was formulated. This problem could then be evaluated by computer allowing for the large number of evaluations needed to compare the algorithm with different parameters and compare the results to other algorithms. In order to make this comparison meaningful one needs to choose a test problem that matches the actual experiment as closely as possible, to do this a n dimensional Gaussian was chosen.

To further increase similarity with the experiment three sources of noise were added: measurement noise affecting the amplitude of the function(σ_a), background noise added to the function(σ_b) and noise in the parameters/position that were passed to the function σ_p.

$$g(x_i, \sigma) = e^{\frac{(x_i^* - x_i)^2}{2\sigma}}$$

$$G(\mathbf{x}, \sigma) = \prod_{i=1}^{n} g(x_i, \sigma)$$

$$f(\mathbf{x}) = \mathcal{N}(1, \sigma_a) * \prod_{i=1}^{n} g(\mathcal{N}(x_i, \sigma_p), \sigma) + \mathcal{N}(0, \sigma_b)$$

As we know only little about the problem landscape of the actual experimental parameters that we are going to optimize this test problem represents a best guess, strictly speaking we cannot induce the suitability to application in the experiment of the algorithm chosen according this test function, however it will be attempted as nonetheless as it is the best starting point available.

4.2 Comparison of Algorithms

Algorithms from different sources were used in the comparison. Two different optimization libraries were used, first the "Parallel Global Multiobjective Optimizer"[4] of which the Python interface is named

PyGMO and secondly the "inspyred"[5] library. Many optimization routines are published only as Matlab code, in these cases the algorithms were applied the the same test problem using Matlab.

In order to compare the algorithms they were all repeatedly applied to the test function using the parameters show in 5.1. The algorithms were run independently 100 times and the following statistics were collected to evaluate the algorithm performance.

Best The median, over all runs, of the best value of each run found so far.

N<-1 The fraction of algorithms that have found a value less than negative one.

Distance The median over all runs, of the squared distance, in parameter space, from the global optimum.

N<.015 The fraction of algorithms that have median squared distance values less that 0.015.

The best value returned so far is the most direct way to measure the performance of the algorithm, for this value, as well as the distance, the median was chosen because it is less susceptible to outliers than the average. In to see how well the population of algorithms we wanted to have a success rate, as returning a value less than -1 was entirely due to noise we chose to count the number of algorithms that had achieved this.

The problem with looking at the fitness value returned is that since it is influenced by noise we may not be finding the actual minimum of the the problem that we are trying to optimize, merely lucky values. To prevent this one can look at the distance from the optimum in coordinate space. First we consider the median, over all runs, distance in coordinate space, despite having considerable jitter this variable still gives a good indication of population state. The jitter results from the median being taken "randomly" from the population, meaning the population diversity is present as well as the diversity between the different runs. A more direct way is counting the number of algorithms where the parameter is within some distance of the optimum. This is done by the $N < .015$ statistic.

4.2.1 Why choose DEA?

In these the three dimensional case4.1 we see that all algorithms converge within the 900 executions that were given. Of the Swarm type algorithms all algorithms have similar performance with the PyGMO PSO algorithm performing best and Bee Colony Optimization performing worst, inters tingly the Bee Colony Optimization does not seem to create any kind of population convergence when looking at the distance statistic. Evolution Strategy type algorithms performed better than Swarm type algorithms. Within this group the inspyred DEA and PyGMO MDE-PBX algorithms performed better than their peers with the inspyred DEA algorithm converging more quickly by about 100 executions. Interestingly N<.015 statistic of the DEA algorithm seems to level off at 0.6, something that does not happen with MDE-PBX, indicating that some population diversity always remains with this algorithm.

As all algorithms converged the optimization algorithms were applied to a 4 dimensional test problem 4.2. The difficulty of the problem increases substantially with each dimension, in a nonlinear fashion, even when one considers only the relative volumes of the search space and test problem Gaussian. Corresponding to this increase in difficulty the number of function evaluations were increased.

For this problem the PyGMO DE, jDE and inspyred PSO algorithms essentially stalled not effusively optimizing. The inspyred DEA and PyGMO MDE-PBX algorithms again performed better than their peers. Interestingly while the inspyred DEA algorithm initially converged than the PyGMO MDE-PBX algorithm the latter caught up and overtook the other at 1000 executions. This corresponds it having better N<.015 values than DEA. In the 4 dimensional case Evolution Strategy type algorithms again out preformed Swarm type algorithms.

Additionally to those tests displayed numerous other one were made. All algorithms shown were also applied to the test problem with 5 and 6 dimensions. For 5 dimensions the DEA algorithm outperformed all others, however did not completely converge within 1000 executions. For the 6 dimensional case all optimization algorithms stalled. As there were a 10 different variants of differential evolution available

for the PyGMO library these were applied to the three dimensional test problem using these, and tuning their parameters, it is possible to achieve performance on par with the inspyred differential evolution, however it is not possible to significantly outperform it. As MDE-PBX comes closest to the performance of the inspyred DEA algorithm it was tested if tuning the parameter of this algorithm caused it to perform better. Tuning the parameter however only increased performance marginally. Finally a similar tuning experiment was done with the Swarm type algorithms, which were found to be fairly insensitive to their configuration parameters as well.

From the test shown, as well as the additional one performed the inspyred DEA algorithm was chosen to be applied to the experiment because of its fast convergence and good performance for all the test problem dimensions evaluated. This algorithm then used to do optimization directly on the experiment.

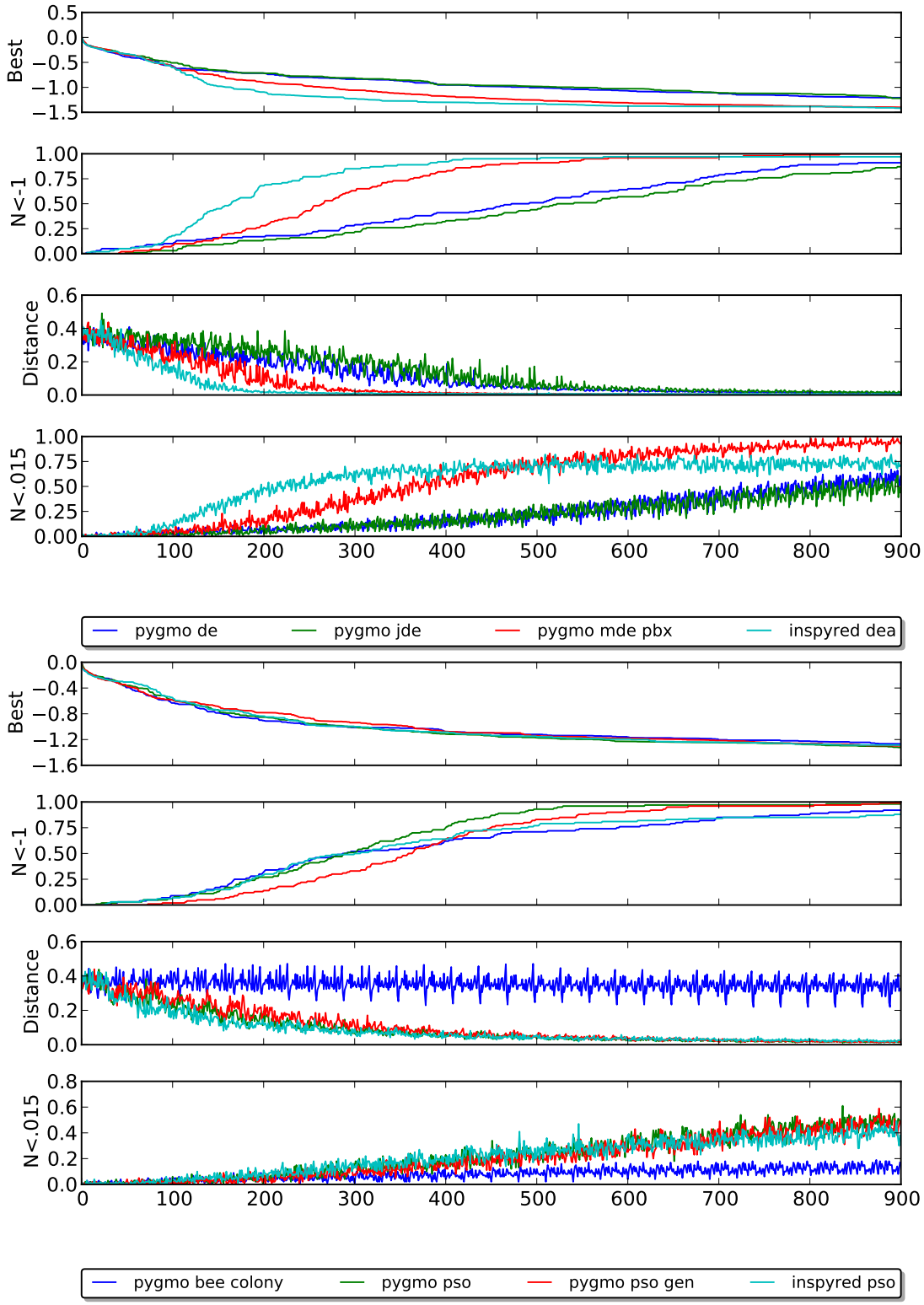


Figure 4.1: Selection of optimization algorithms applied to 3 dimensional test problem. Evolution Strategy and Swarm type algorithms shown on top and bottom respectively.

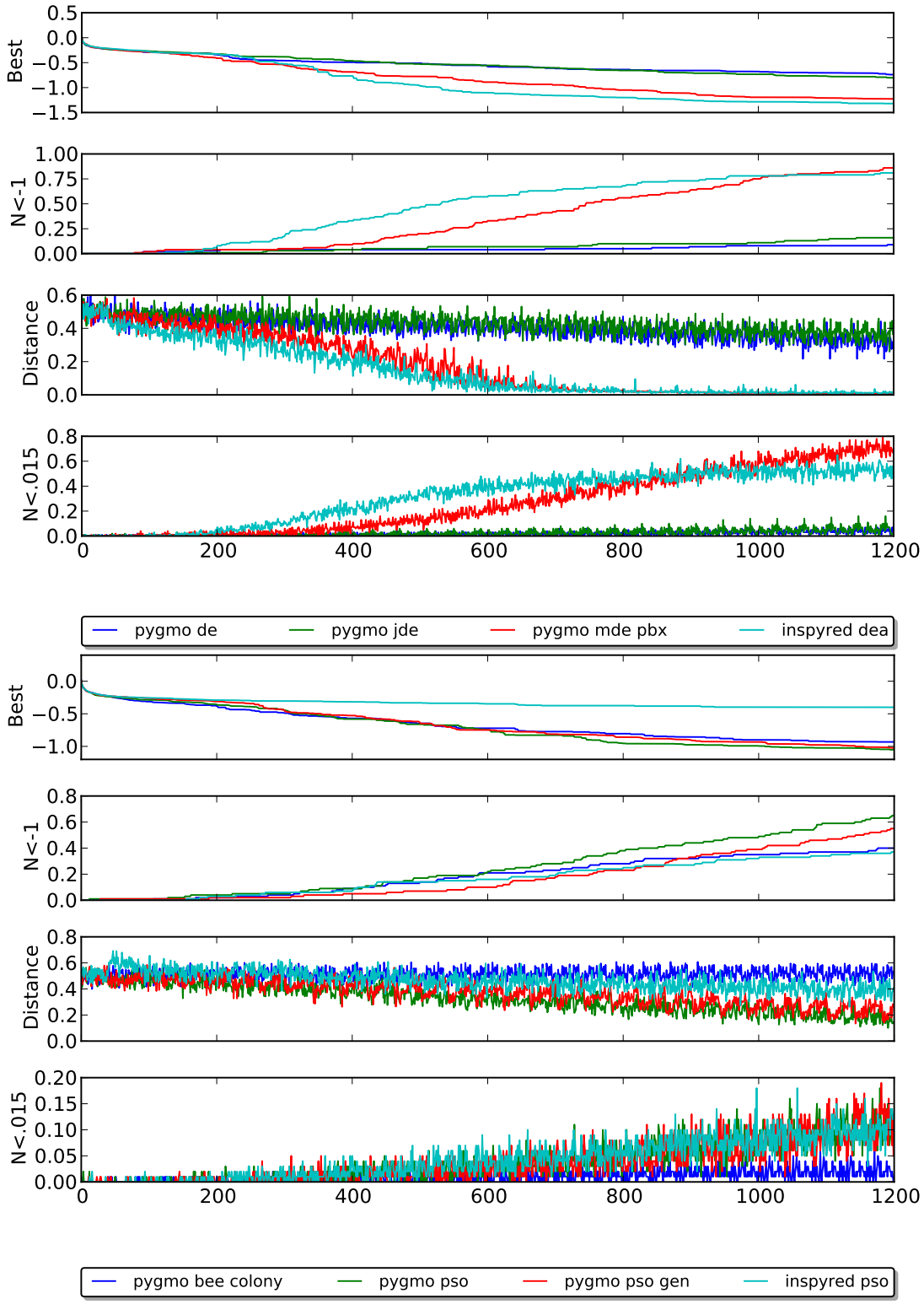


Figure 4.2: Selection of optimization algorithms applied to 4 dimensional test problem.Evolution Strategy and Swarm type algorithms shown on top and bottom respectively.

Chapter 5

Appendix

5.1 Test Problem Parameters

Unless otherwise specified the optimization was run using the following parameters.

Problem Gaussian width $\sigma = 0.1$

Position noise $\sigma_p = 0.05$.

Amplitude noise $\sigma_a = 0.2$.

Background noise $\sigma_d = 0.1$.

Dimension $n = 3$ $n \in [0, 1, \ldots, 6]$

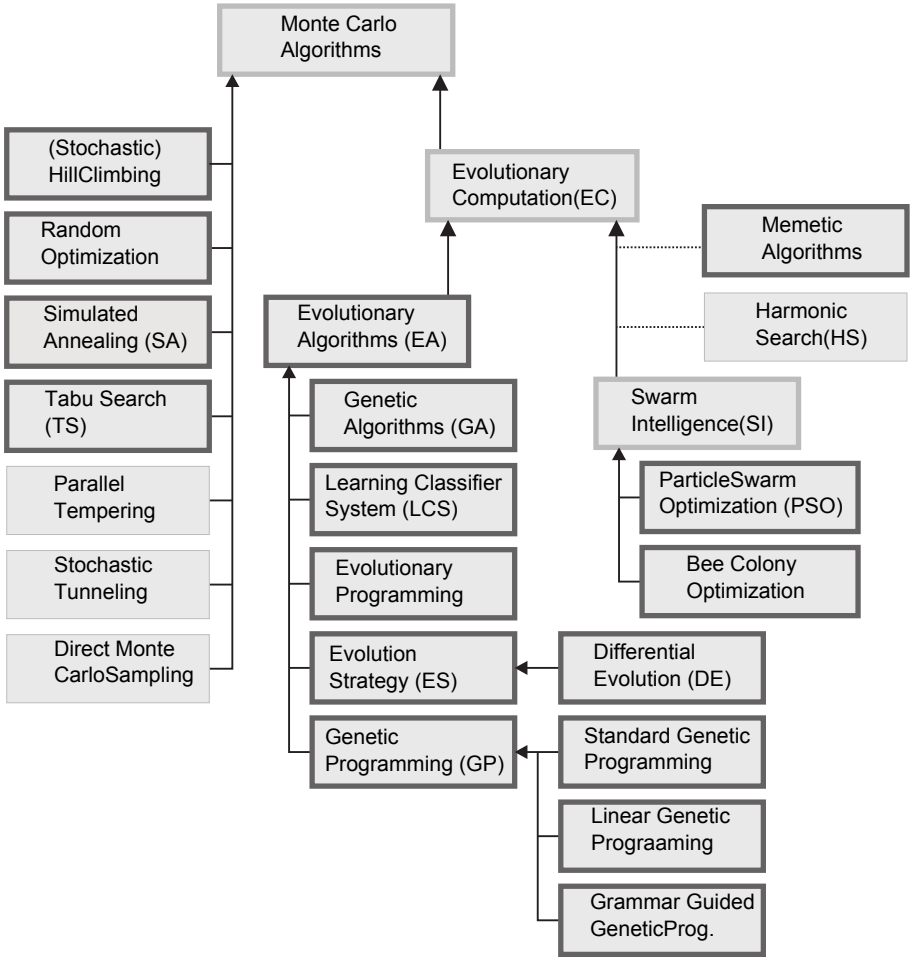
Bounds $x_i \in [0, 1]$ $\forall i \in \{0, \ldots, n\}$

5.2 List of Optimization Algorithms Considered

A list of all the algorithms that were applied to the test problem. Those not included in the comparison were omitted because they performed badly.

Algorithm Name	Library	Reference
Differential Evolution (DE)	PyGMO, inspyred	
Self-adaptive DE (jDE)	PyGMO	
DE with p-best crossover (mde-pbx)	PyGMO	
Simple Genetic Algorithm (SGA)	Matlab	[6]
Estimation of Distribution (EDA)	inspyred	
Particle Swarm Optimization (PSO)	PyGMO, inspyred, Matlab	[7]
Bee Colony Optimization	PyGMO	
Improved Harmony Search (IHS)	PyGMO	
Pattern Search	Matlab	[6]
Simulated Annealing (SA)	PyGMO, inspyred, Matlab	[6]
Multi-Lever Coordinate Search (LCS)	Matlab	
Local Uni-modal Sampling (LUS)	Matlab	[8]
Covariance Matrix Adaptation Evolution Strategy (CMA-ES)	Matlab	

5.3 Classification of Optimization Algorithms



Bibliography

[1] Thomas Weise, *Global Optimization Algorithms, Theory and Application.* 2nd Edition, 2009 http://it-weise.de.

[2] Brian Hayes, *A slight discrepancy.* http://bit-player.org/2011/a-slight-discrepancy http://devio.us/ tripzilch/raindrops/ 2011.

[3] Rainer Storn and Kenneth Price, *Differential Evolution - A Simple and Efficent Heuristic for Global Optimization over Continous Spaces.* Journal of Global Optimization, 1997

[4] Izzo, D. *PyGMO and PyKEP: Open Source Tools for Massively Parallel Optimization in Astrodynamics* International Conference on Astrodynamics Tools and Techniques, 2012.

[5] Aaron Lee Garrett *inspyred: Bio-inspired Algorithms in Python* inspyred 1.0 http://inspyred.github.com/ 2012

[6] Thomas Coleman, Mary Ann Branch and Andy Grace, *Optimization Toolbox for use with Matlab User's Guide* Version 2, 2002 The Math Works, Inc. 24 Prime Park Way, Natick, MA 01760-1500.

[7] M. Clerc, J. Kennedy *"The particle swarm - explosion, stability, and convergence in a multidimensional complex space* Feb 2002 Evolutionary Computation, IEEE Transactions on , vol.6, no.1, pp.58-73, doi: 10.1109/4235.985692

[8] M.E.H. Pedersen, *Tuning & Simplifying Heuristical Optimization* 2010. PhD Thesis University of Southampton