CS7461 Machine Learning: Assignment 2

Matthias Grundmann
grundman@cs.tum.edu

March 9, 2007

1 Optimization problems

I have chosen 3 different optimization problems to demonstrate the various strengths of each algorithm. I picked a variation of Rastrigin's function, the optimization of the Travel-Salesman-Problem (TSP) and the One-Max-Function. Each of them and the motivation behind them will be explained in the following. Please note, that the first two functions are minimization problems, which can be easily transformed to maximization problems by changing the sign of the cost or the fitness function, so this is not a restriction.

1.1 Rastrigin's function

Rastrigin's function[5] is basically the addition of a periodic function and parabola over $\mathbb{R}^2$. Although this is basically a continuous problem, the representation of floats as 32-bit binary vectors transforms this to discreet problem, so that transformations like cross-over and mutation can be applied. I used the variation of Rastrigin's given by $(x, y) \mapsto 20 + 3(x^2 + y^2) - 10(\cos(2\pi x) + \cos(2\pi y))$. The plots in 1 highlight the difficulty of the function.

The function has infinite local minima, and only one global minima at $(0, 0)$. So it is very likely for an optimizer to fall in one of the local minima. Especially algorithms that rely solely on gradient descent methods and start from a single point are expected to perform very bad. The function is one example of many artificial functions[1] created to evaluate the performance of different optimization algorithms. Although the function is academical with respect to the periodicity of the local minima, an algorithm that performs well on this function is expected to perform well on real world problems where many local minima occur, i.e. in fluid dynamics as shown in [4].

[1] for a comprehensive overview see [1]

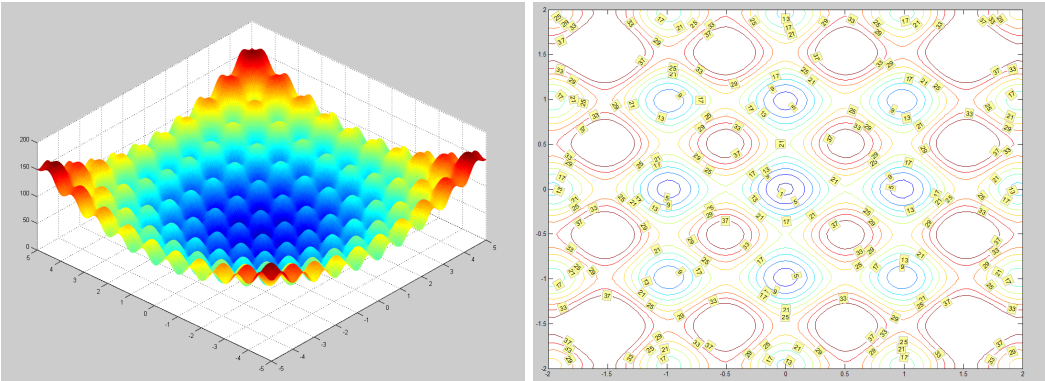

Figure 1: Rastrigin's function. Left: surface plot - Right: contour plot

1.2 TSP

The well known Traveling Salesman Problem is proven to be NP-hard. The goal is to find the shortest round-trip between n cities while visiting each city just once. TSP problems also occur in every day life. Planning optimal routes between cities is a crucial task for logistic companies or a backpacker who wants to visit some landmarks. Although in this cases n may be rather small. Other applications may include factory scheduling, wiring looms and circuit board drilling. I will focus here on the special case of a symmetric euclidian TSP, which is still NP-hard. Given n cities and picking an arbitrary city to start from, there exists $\frac{(n-1)!}{2}$ possibilities for a round-trip. The factor 2 arises from the fact, that we don't care about the direction in which we are traveling. For this assignment I try to find the round-trip between the capitals of the $n = 48$ US states(exclusive Alaska and Hawaii). See figure 2 for the point configuration. This small example already leads to approximately $1.3 \cdot 10^{59}$ possible routes. This example is taken from [2] and has a proven optimal route of 33523.7 mi. The optimal route is also shown in figure 2. Besides the fact that the number of possible routes grow exponentially with respect to the number of cities, the TSP problem is not 'discreet continuous'[2]. That means that small changes in the configuration of the points can lead to completely different optimal routes. That makes it hard to define a good search algorithm, and a greedy algorithm is very unlikely to find the optimal route.

1.3 One-max

The One-max function is a discrete function defined for a bit-vector x of length n , that tries to maximize $\sum_{i=0}^{n} x_i$. The maximum will be obtained if $x_i = 1$ for all i. Although the search direction may be obvious for the human, the function oscillates highly with

[2]This term is just mathematically inspired but not clearly defined

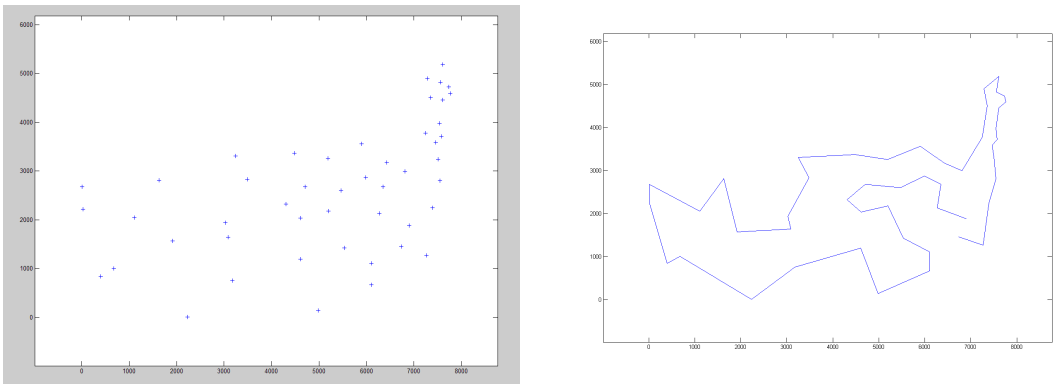

Figure 2: TSP problem. Left: 48 capitals of US states - Right: optimal route

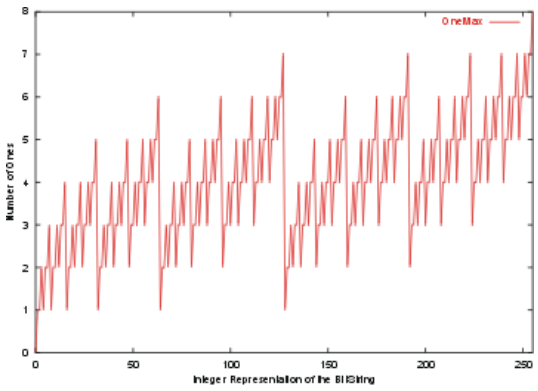

Figure 3: One-Max function

respect to the integer-representation. Figure 3 shows the function and was obtained from the internet. Note that one-max is 'discreet continuous'. Small changes in x (flipping a bit) will lead to small changes in the results (± 1).

2 Optimization Algorithms

The algorithms that are compared are:

Randomized Hill-Climbing (RHC) The MATLAB function **patternsearch** was used, and called from random start points. **patternsearch** implements the normal hill-climbing algorithm and additional a scheme that increases or decreases the search radius in every iteration-step based on whether there exists a neighbor with a better fitness function value. Two different polling schemes were evaluated. The first stops

an iteration when a neighbor with a better fitness function values is found, the second evaluates all neighbors and take the best one (complete polling).

Genetic Algorithm (GA) The MATLAB function **ga** was used and called with different population sizes. The initial parameters for mutation (gaussian noise) and crossover (80% percent of the old population replaced by children) were used. The best 2 individuals are kept among generations.

Simulated annealing (SA) Simulated annealing was found to be sensitive to the initial temperature and the perturbation functions used. An exhaustive search for the best parameters like for SVMs could be used, but would take a significant amount of time and was considered to be beyond the scope of the assignment. The values were adapted by try and error. It has been found that not the absolute values for initial temperature and the amount of perturbation were significant but their ratio.

MIMIC The implementation from ABAGAIL was used. The population size was chosen according to [3] to 200 and 100 iterations were used by default.

Cause all algorithms are randomly initialized, running an algorithm once on the data and try to derive conclusions about the nature of the algorithm is not suited. Instead all algorithms were run several times and the mean was used for comparison. The specific iterations are given in the results.

3 Results

This section gives details about the parameters used for optimization as well as the results of the different algorithms.

4 Rastrigin's function

All algorithms were run 20 times, except MIMIC which was run 50 times. In addition the GA was run for varying population sizes varying from 5 to 50. The average results are given in in table 1. The error is the distance to the true minimal function value of 0. All initial points were generated on the interval $[-20, 20]x[-20, 20]$. Clearly RHC and SA outperform the other algorithms, both have nearly no error. Within RHC there is no significant difference between first and complete poll. However SA performs better than RHC, both in wall clock time (20%) and in function evaluations (43.7 %). Cause of the many local minima only algorithms that can examine points far away from the current best point have a chance to find the global minima. It is essential that this property remains during execution. SA 'jumps' around with high temperature and cools down slowly. So it is very likely to 'hit' the right region that will lead it to the minima. Ordinary RHC will in general only find the global minima, when one of the initial points is placed in the

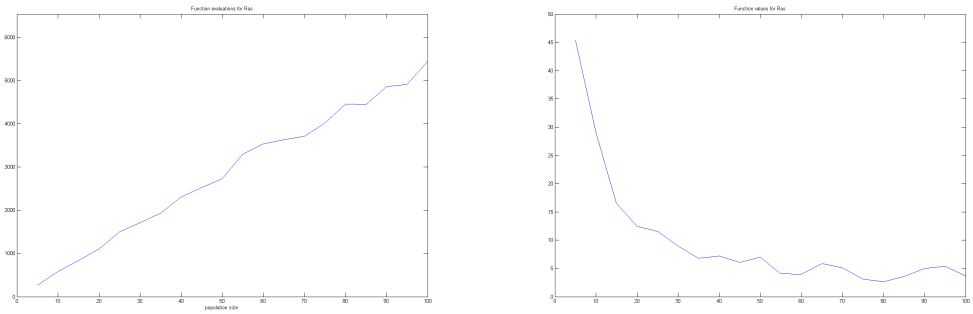

Figure 4: GA for Rastrigin's function. Left: Error for different populations - Right: Function evaluations

right region at initialization, cause it examine only points in the direct neighborhood of the current point. This showed a separate test, where outlier occurred. However in the implementation I used, the expansion of the search radius enables the algorithm to examine points far away and to find the global minima. GA performs pretty bad. However the error decreases significantly with increasing population size, [3] although the function evaluations increases also. This is shown in figure 4. In the contour plot 5 it is shown, that the GA often tends to fall in local minima. Compared to MIMIC one notice, that MIMIC when falling into local minima, finds the local minimum very precise, while the optima GA found seem to be scattered around the minima. A reason might be, that MIMIC interpolates the probability distribution and Rastrigin's function is smooth (infinite differentiable), so it will find the minima. On the other hand GA uses non-linear bit-transformations to mutate the individuals of a population, which does not model the smoothness of Rastrigin's function.

Algorithm	optimization time s	function evaluations	error s
RHC - first poll	20.5199	29684	0
RHC - complete poll	18.7099	29396	0
GA	27.3438	2892.7	9.6744
SA	5.476	12976.6	0.001101
MIMIC	179	20000	1.1832

Table 1: Rastrigin's Function Results

[3] only 2.2138 with 100 individuals

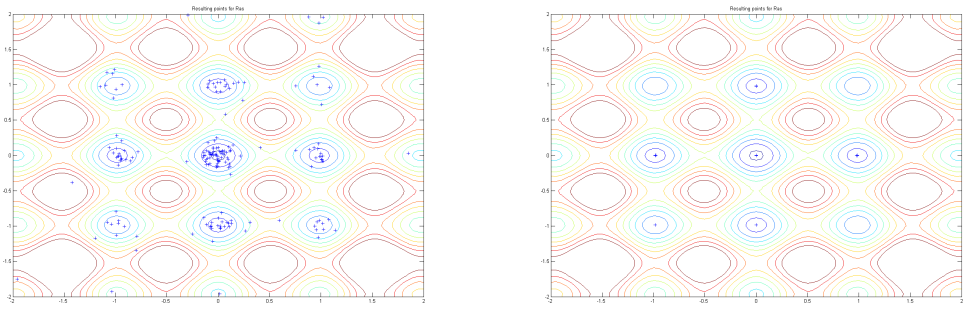

Figure 5: Best points for Rastrigin's function - Left: GA - Right: MIMIC

5 TSP

All algorithms were run 10 times, except MIMIC which was run 5 times. In addition the GA was run for varying population sizes varying from 5 to 50. The average results are given in in table 1. The error is the distance to the true minimal function value given by the optimal tour. A tour was represented by the permutation that one obtain when sorting a vector. This enables to use ordinary addition and cross-over operators, while calculating the neighbor or the cross-over of two permutations is not trivial. One could argue, that this may lead to bad results, but it has found in a separate test, that substituting the neighbor operation by swapping two cities does not improve the optimization results. One can think of many definitions of a neighbor of a permutation and they are all equally well. Clearly all algorithms perform not very good, the route they compute is in average 50 % too long. Like with Rastrigin's function GA improves with increasing population size, therefore the charts are omitted. Unlike with Rastrigin's function the TSP problem is not smooth and hard to analyze. It seems, that due to its 'discontinuities' [4] MIMIC and GA perform bad. Interpolation among them may fool the algorithms. It should be noted, that there exists TSP-optimized variants that implement more sophisticated cross-over and mutation operations. In my representation a mutation might not change the route at all. RHC and SA perform equally well, although this time the complete poll strategy for RHC leads to a 1000 mi better route in average. I think both algorithms perform pretty good, cause they explore a huge search space and are likely to find good routes. The best obtained route for SA and GA are given in figure 6. Although RHC performs equally well to SA, it is way slower. Also RHC expands its search radius, it stays in its current neighborhood and searches within it. The restart from a different random point may change this. SA evaluates not only points in the current neighborhood but also other points. That might be why it finds a 'good' route faster than RHC.

[4] see above for justification of the use of this word

Algorithm	optimization time s	function evaluations	error s
RHC - first poll	3472.1	4614.8	17421.7
RHC - complete poll	3367.1	4515.6	16414.37
GA	690.8	3937.95	47786.9
SA	128.8	6688.5	16857.1
MIMIC	331	100000	32566.82

Table 2: Rastrigin's Function Results

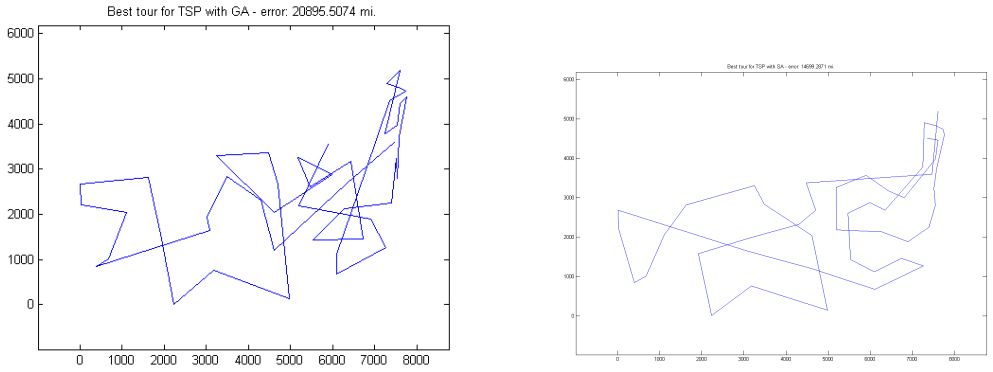

Figure 6: Best routes obtained for TSP problem. Left: GA Right: SA

6 One-Max

The optimization of One-Max was done with the ABIGAIL package and the average results over 100 runs (MIMIC 10 runs) are given in table **??**. n is set to 80, so the maximum should be obtained at 80.

Algorithm	optimization time s	function evaluations	value s
RHC	0.07	20000	68.73
GA	0.06	20000	60.93
SA	4.3	60000	48.21
MIMIC	15.9	50000	76.9

Table 3: One-Max Function Results

In this case SA performs worst, although it evaluates many points. The reason is that one-max relies on a search scheme that interpolates evaluated points, that is why MIMIC excels all other optimization algorithms, which only check neighbors. As can be seen in figure 3 this will often lead to local minima or worse configurations. I think MIMIC most probably performs better that the other algorithms, cause One-Max is 'discrete continuous', so an interpolation will seriously guide the search towards the optimum value.

7 The neural network

I used the neural network from assignment one to recognize cancer cells from features extracted from microscopic images. I showed that a network with 1 hidden layer of 7 units can achieve classification rates of 97.8 %. I used the standard minimization of the sum of squared distances (SSD) of the error with regularization term to prevent the solution from blow-up, that might be occur during RHC. The regularization term is just the addition of the norm of weights of the units. With a 20 dimensional input, that leads to a $20 * 7 + 7 * 1 = 147$ dimensional problem. The weights were initialized randomly and each optimization algorithm was run 10 times. The mean results are given in table **??**. The recognition rate was obtained by applying the network to a separate test set. There are some interesting results to notice. First the function value we try to minimize does not lead necessarily to a good recognition rate. This may have two reasons. First the introduction of the regularization term and second overfitting may occur, when try to minimize the SSD without any cross-validation set. It is also interesting that the GA performs best in this case. It might be that mutation and especially cross over model structural changes of connections of the layers and therefore lead to very good results. As usual RHC needs the most function evaluation, followed by GA and SA.

Algorithm	optim. time s	fun-evals	average fun-value	best value	best recogn. rate
RHC - first poll	1290.96	5034.2	7.3685	6.6078	0.76842
GA	916.02	3420	14.7817	10.6755	0.87895
SA	498.84	3958	10.5211	9.3258	0.71053

Table 4: Neural Network Training Results

8 Conclusion

It could be shown that smooth problems can be solved best with SA (Rastrigin's function and neural network training), while 'discreet smooth' problems like One-Max are best suited for MIMIC. RHC is an exhaustive search which performs very well on smooth problems, but has serious difficulties with discontinuities. All 4 algorithms does not perform well on problems like TSPs, where special variants of them are needed to obtain good results. If we have a smooth problem with a global minima, like in the case of minimizing the SSD in case of the NN, none of the algorithms can outperform an ordinary Gradient descent method.

References

[1] http://www.geatbx.com/docu/fcnindex-01.html.

[2] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

[3] J. S. de Bonet, C. L. Isbell, Jr., and P. Viola. MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 424. The MIT Press, 1997. Available from: citeseer.ist.psu.edu/debonet96mimic.html.

[4] F. Muyl, L. Dumas, and V. Herbert. Hybrid method for aerodynamic shape optimization in automotive industry. *Computers and Fluids*, 33(5):849–858, June 2004. Available from: http://dx.doi.org/10.1016/j.compfluid.2003.06.007.

[5] A. Törn and A. Zilinskas. Global Optimization. *Lecture Notes in Computer Science*, 350, 1989.