ON THE EFFECT OF STEP WIDTH SELECTION SCHEMES ON THE PERFORMANCE OF STOCHASTIC LOCAL SEARCH STRATEGIES

Lars Nolle
School of Computing and Mathematics
The Nottingham Trent University
Burton Street, Nottingham, NG1 4BU, UK
lars.nolle@ntu.ac.uk

KEYWORDS

Optimisation, Hill Climbing, Step Width Selection.

ABSTRACT

This paper examines and discusses the effects of different step width selection schemes on the effectiveness of stochastic local search strategies. It is shown that the success of these search strategies depends, among other things, heavily on the chosen neighbourhood definition for a particular application. Simulations have shown that the use of randomly selected steps with a defined upper limit increases the probability of finding the global optimum, in contrast of using a constant step width. It has also been demonstrated that the efficiency of stochastic local search strategies can be improved by reducing the maximum step width over time.

INTRODUCTION

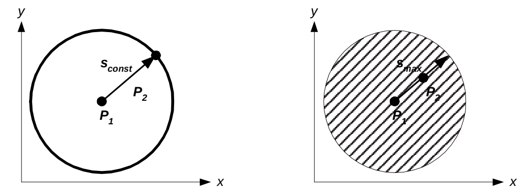
Stochastic local search algorithms, like Hill-Climbing (Hopgood 2001) or Simulated Annealing (Metropolis et. al. 1953) (Kirkpatrick et. al. 1983), are a class of search strategies that start from a random point in the search space and that test new potential candidate solutions that are randomly selected from the 'neighbourhood' of the current solution. If the candidate solution has a higher fitness than the current solution it replaces the current solution.

For continuous parameter optimisation, it is practically impossible to choose direct neighbours, because of the vast number of points in the search space. In this case, it is necessary to choose new candidate solutions from a wider neighbourhood, i.e. from some distance of the current solution, in order to travel in an acceptable time through the search space.

In principle, there are two ways to traverse through the search space, firstly using equidistant steps and secondly, using steps of random length.

Equidistant vs. Random Length Steps

The distance between the current solution P_1 and a candidate solution P_2 could either be a fixed step width s_{const} or it could have an upper limit s_{max}. In the first case, the neighborhood would be defined as the surface of a hypersphere around the current solution P_1 (Figure 1a), in the second case the neighborhood would be defined as the volume of the hypersphere and hence a candidate solution would be drawn randomly from within these hypersphere (Figure 1b).

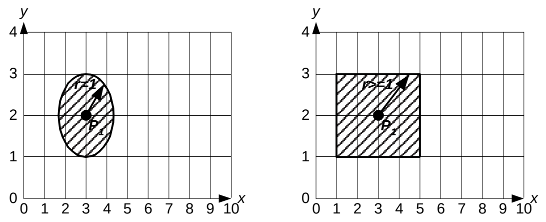


a) Constant Step Width   b) Maximum Step Width

Figure 1: Possible Neighbourhood Definitions

Vector vs. Component Selection

In case of a constant step width, the displacement vector between P_1 and P_2 needs to be calculated for each step. One problem associated with this calculation is, that for dimensions of the input space with unequal ranges, the neighbourhood becomes distorted (Figure 2a). As a consequence to that, the input space will be sampled with a higher resolution in one dimension compared with the other one. To overcome this problem, one could determine the displacement for each input dimension separately using a scaling factor to compensate for the different resolution (Figure 2b).



a) Distortion   b) Compensated

Figure 2: Effect of Distortion Due To Unequal Resolutions of Input Space Dimensions and the Compensation by Component Selection

Aim of Research

The aim of this work was not to compare different search algorithms in order to find the most efficient one to solve a set of test functions. Instead, the aim was to examine the effects of different step width selection schemes on the effectiveness of stochastic local search strategies. Therefore, the basic hill-climbing algorithm was used to find the minimum of two different test functions, a high dimensional one and a multi-modal one.

TEST FUNCTIONS USED

For the simulations, two well established test functions were used, DeJong's first test function (DeJong 1975) and Schwefel's function (Schwefel 1981). Figure 3 shows a graphical representation of a two dimensional version of DeJong's 1st function, whereas Figure 4 depicts a two dimensional version of Schwefel's function.

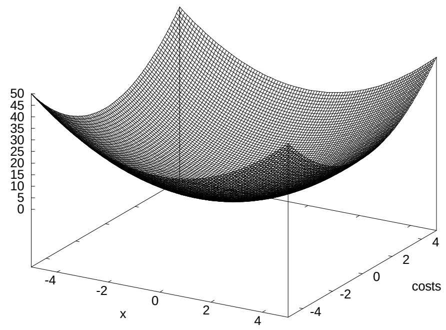


Figure 3 – DeJong's 1st Function (2D)

DeJong's 1st function, also known as sphere model, has a minimum cost value of zero within the defined range [-5,+5] for each input dimension. For the experiments a 10 dimensional version was used.

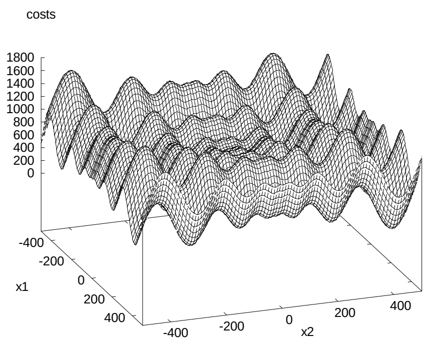


Figure 4 –Schwefel's Function (2D)

Schwefel's function has also a minimum cost value of zero within the defined range [-500,+500] for each input dimension. In this research a two dimensional version was used.

EQUIDISTANT VS. RANDOM LENGTH STEPS

In this set of experiments, the effect of using a random step width instead of a fixed step length has been examined. For both test functions, the step width, i.e. either constant or random, was varied and 1000 simulations per step width were carried out.

DeJong's 1st Function

For the 10 dimensional version of DeJong's 1st function, 2000 iterations were allowed per run. Figure 5 shows the results of the simulations. The dots represent the average costs achieved during the experiments and the error bars represent the standard deviation achieved.

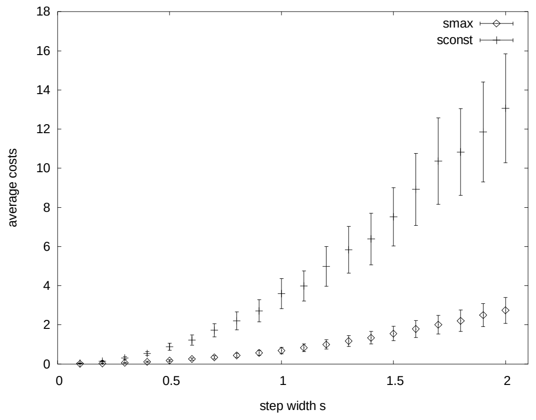


Figure 5: Average Costs vs. Step Width for DeJong's 1st Test Function

It can be seen that the average costs achieved using the s_{max} selection scheme where significant lower compared to the s_{const} selection scheme. On the other hand, Figure 6 shows that it took the s_{max} selection scheme longer to converge.

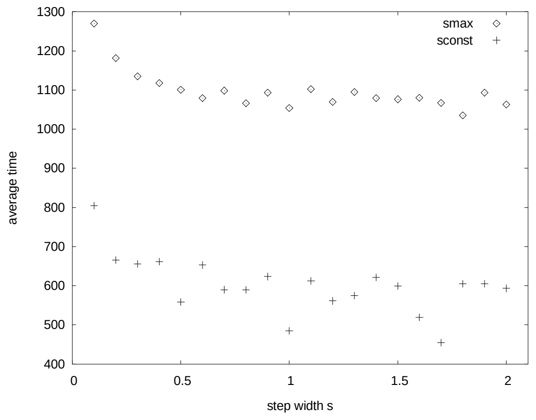


Figure 6: Average Time in Iterations for Finding the Best Solution During a Search for DeJong's Function

However, this is not a disadvantage, this only indicates that the s_{const} selection scheme simply failed to further improve the solutions found whereas the s_{max} selection scheme achieved a constant improvement even towards the end of the search runs.

Schwefel's Function

For the two dimensional version of Schwefel's function, 2000 iterations were allowed per run. Figure 7 shows the results of the simulations. The dots represent the average costs achieved during the experiments and the error bars represent the standard deviation achieved.

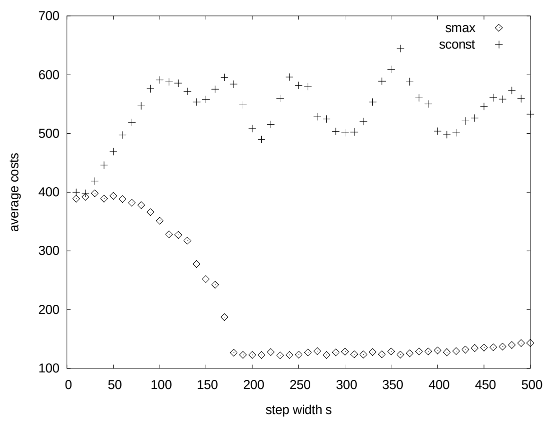


Figure 7: Average Costs vs. Step Width for Schwefel's Test Function

It can be seen from Figure 7 that small values for s_{max} result in relatively high costs, similar to the costs achieved using the s_{const} selection scheme. But these costs dropped dramatically when s_{max} reached 180 in this experiments, whereas the average costs achieved using s_{const} increased with increasing values for s_{const}.

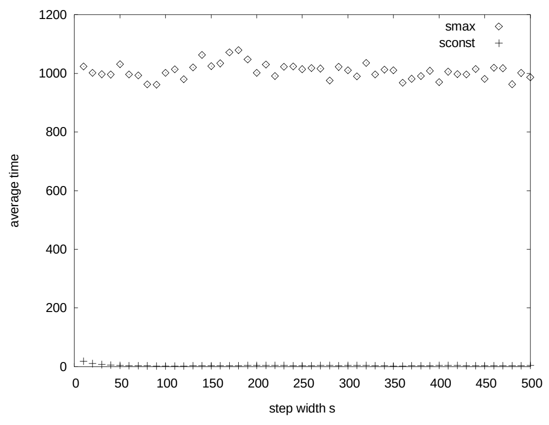


Figure 8: Average Time in Iterations for Finding the Best Solution During a Search for Schwefel's Function

Figure 8 shows the average time (in iterations) that was needed to reach the optimum for both selection schemes. Again, it can be seen that the s_{max} scheme found its best solutions after approximately 1000 iterations, in contrast to s_{const}, which converged significantly faster.

Discussion

Figure 9-Figure 13 show steps in typical search runs using the different selection schemes to minimise Schwefel's function. Each dot represents an accepted solution, i.e. the rejected solutions are not shown. The starting point for all experiments is on top of the local hill at about –200. Figure 9 and Figure 10 show the results for using a constant step width, Figure 11, Figure 12 and Figure 13 show the results for experiments using a maximum random step width.

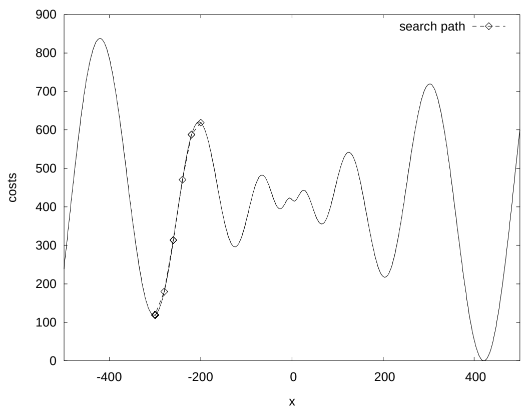


Figure 9: Constant Step Width s_c = 20

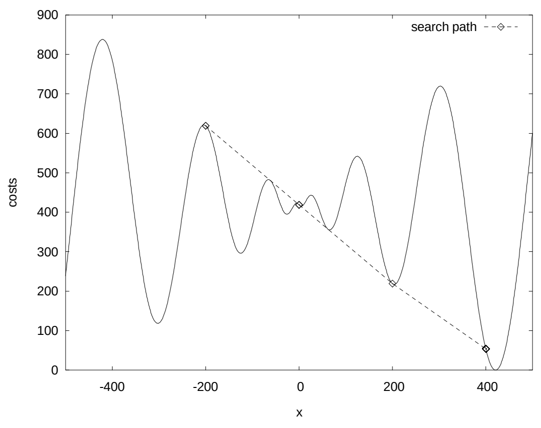


Figure 10: Constant Step Width s_c = 200

As it can be seen from Figure 9, if s_{const} is chosen to be too small, the algorithm gets trapped in an adjacent local optimum. If s_{const} is large enough, it is capable of jumping over local optima (Figure 10), but because the steps are of constant length, the number of points, which can be reached from a certain position, is limited. For example, in Figure 10, it is not possible to jump from the last point (near +400) to a point with lower associated costs, because all these points are within the constant step width.

When using a random step length s_{max}, the effect for choosing small values is similar to that for using a constant step width (Figure 11), but with increasing s_{max}, the ability to reach points with a better cost value increases. In Figure 12 for example, the algorithm was able to jump out of the valley on the right hand side of the start point into the valley of the left hand side, and it was also able to decent into the valley on the left hand side. However, because the maximum step width was not large enough to jump into the right hand side of the search space, which contains the global minimum, the algorithm only exploited the local optimum. In Figure 13, the maximum step length is large enough to jump into the right hand side of the search space, but this time, because of the large steps that are possible, it jumps back into a local optimum.

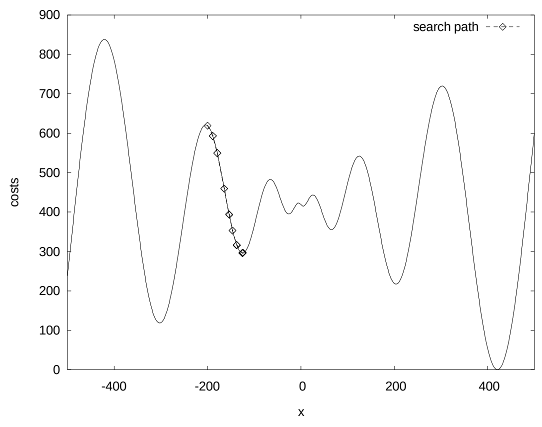


Figure 11: Maximum Step Width s_{max} = 20

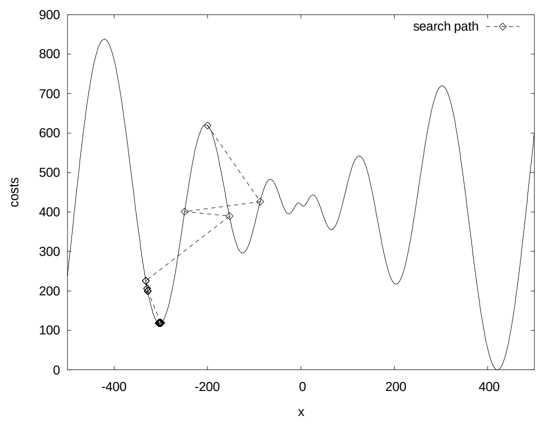


Figure 12: Maximum Step Width s_{max} = 200

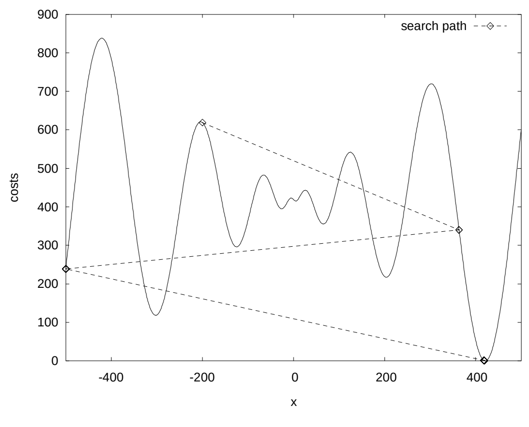


Figure 13: Maximum Step Width s_{max} = 1000

Conclusion

The results given above show, that, if the step width is chosen to be too small, both selection schemes will eventually get trapped in the nearest local optimum, whereas using a larger step width increases the ability to overcome local optima. However, for a constant step width it is more difficult to reach the global optimum, because of the limited number of possible search points that can be reached from a certain position.

This is overcome by using a maximum random step length, but this needs to be carefully selected: if it is chosen to be too large, the algorithm will lose its ability to exploit a potential region, which can also be seen in Figure 5, which documents the results for DeJong's 1st function, which only has one optimum.

But if the step width is chosen to be too small, it is not guaranteed that the algorithm can jump over local optima in order to reach the region that contains the global optimum. Therefore, the careful selection of appropriate values for s_{max} seems to be crucial for the success of the algorithm for a particular application.

REDUCING S_{MAX} OVER TIME

The findings above indicated that a large value for s_{max} supports exploration of the search space, whereas small numbers for s_{max} increase the exploitation capabilities of the search algorithm. It would be advantageous if an algorithm would explore the search space at the beginning of a search and exploit the most promising region in the later stages of the search.

Therefore, in the next set of simulations, three different scaling functions have been used to subsequently reduce s_{max} during the run, starting with a relatively large start value. Equation (1) achieves a linear reduction, equations (2) and (3) reduce s_{max} exponentially. Figure 14 shows a graphical representation of the three functions.

$$s_{\max}(i) = LIN(i) = \frac{-s_0}{i_{\max}} i + s_0 \tag{1}$$

$$s_{\max}(i) = EXP1(i) = s_0 \cdot e^{\frac{-\alpha}{i_{\max}} i} \tag{2}$$

$$s_{\max}(i) = EXP2(i) = s_0 + s_0 \frac{1 - e^{\alpha i / i_{\max}}}{1 - e^{-\alpha}} \tag{3}$$

Where:

i:	iteration
i_{max}:	maximum number of iterations
s_0:	initial maximum step width
$s_{max}(i)$:	maximum step width at iteration I
α:	constant, determines the degree of non-linearity

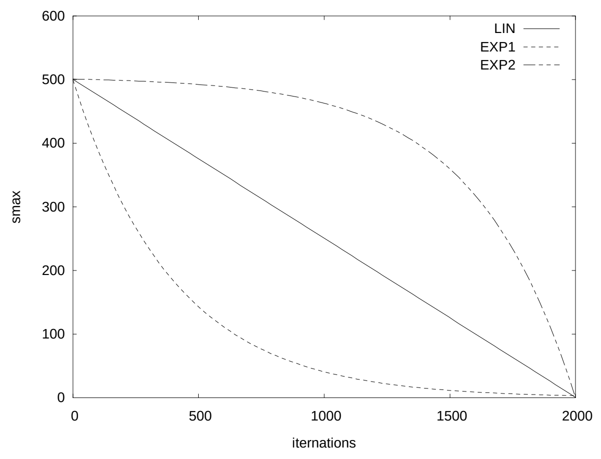


Figure 14: Scaling Functions Used

Each scaling function was used 1000 times and the results are compared with the best results obtained using a non-scaled maximum step width (see Figure 7).

Comparison of Results

Table 1 shows the result of the experiments for the non-scaling algorithm and the three algorithms using scaling functions, in Table 2 the performance of the algorithms are compared with the standard non-scaling one.

Table 1 – Results

	Constant	**Linear**	**Exp1**	**Exp2**
Average costs	132.90	124.29	121.99	125.22
Standard deviation	83.38	85.17	87.98	86.13
Average time	1105.9	1984.3	1800.3	1981.9
Standard deviation	578.0	69.9	211.8	117.1

Table 2 – Improvements by Algorithms

	Linear	**Exp1**	**Exp2**
Average costs	6.48%	8.21%	5.78%
Standard deviation	-2.15%	-5.52%	-3.30%
Average time	-79.43%	-62.79%	-79.21%
Standard deviation	87.91%	63.36%	79.74%

It can be seen that by using the scaling function EXP1 the average costs could be further reduced by up to 8.21%. The standard deviation of the average costs, on the other hand, increased by 5.52%. This, and the fact that the average time until the algorithm found the best solution increased by 62.79%, is probably an indication that the algorithm converges slower because of the – now reduced – speed with that the algorithm is traversing over the fitness landscape.

CONCUSIONS

In this work it was shown that the definition of the neighbourhood is crucial to the success of the algorithm. Two different selection schemes are compared in this work: equidistant steps and random steps with an upper limit. It has been demonstrated that random steps with an upper limit outperform neighbourhood selection schemes with a constant step length.

It was also shown that using a scaling function to reduce the maximum step with over time could again increase the performance of the algorithm.

AUTHOR BIOGRAPHY

Lars Nolle graduated from the University of Applied Science and Arts in Hanover in 1995 with a degree in Computer Science and Electronics. After receiving his PhD in Applied Computational Intelligence from The Open University, he worked as a System Engineer for EDS. He returned to The Open University as a Research Fellow in 2000. He joined The Nottingham Trent University as a Senior Lecturer in Computing in February 2002. His research interests include: applied computational intelligence, distributed systems, expert systems, optimisation and control of technical processes

REFERENCES

Hopgood, A.A. 2001. “Intelligent Systems for Engineers and Scientists”. 2nd ed., CRC Press

Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P. 1983 “Optimization by Simulated Annealing”. *Science*, Vol. 220, No. 4598, 671-680

Metropolis, A., Rosenbluth, W., Rosenbluth, M. N., Teller, H., Teller, E. 1953. “Equation of State Calculations by Fast Computing Machines”. *Journal of Chemical Physics*, Vol. 21, No. 6, 1087-1092

DeJong, K.A. 1975. “An analysis of the behavior of a class of genetic adaptive systems”. PhD Thesis, University of Michigan

Schwefel, H.-P. 1981. “Numerical optimization of computer models”. Wiley and Sons