Etschberger, Stefan; Hilbert, Andreas

Working Paper

Multidimensional Scaling and Genetic Algorithms : A Solution Approach to Avoid Local Minima

Arbeitspapiere zur mathematischen Wirtschaftsforschung, No. 181

Provided in Cooperation with:
Universität Augsburg, Institut für Statistik und Mathematische Wirtschaftstheorie

Suggested Citation: Etschberger, Stefan; Hilbert, Andreas (2002) : Multidimensional Scaling and Genetic Algorithms : A Solution Approach to Avoid Local Minima, Arbeitspapiere zur mathematischen Wirtschaftsforschung, No. 181, Universität Augsburg, Institut für Statistik und Mathematische Wirtschaftstheorie, Augsburg

This Version is available at:
https://hdl.handle.net/10419/22817

ARBEITSPAPIERE ZUR
MATHEMATISCHEN WIRTSCHAFTSFORSCHUNG

Multidimensional Scaling and Genetic Algorithms: A Solution Approach to Avoid Local Minima

Stefan Etschberger and Andreas Hilbert

Heft 181/2002

Institut für Statistik und Mathematische Wirtschaftstheorie
Universität Augsburg

Stefan Etschberger
Universitätsstr. 16
D - 86159 Augsburg
Telefon +49 821 598-4159
Telefax +49 821 598-4226
EMail Stefan.Etschberger@wiwi.uni-augsburg.de

Dr. Andreas Hilbert
Universitätsstr. 16
D - 86159 Augsburg
Telefon +49 821 598-4155
Telefax +49 821 598-4226
EMail Andreas.Hilbert@wiwi.uni-augsburg.de

Multidimensional Scaling and Genetic Algorithms: A Solution Approach to Avoid Local Minima

Stefan Etschberger and Andreas Hilbert

April 10, 2003

Abstract

Multidimensional scaling is very common in exploratory data analysis. It is mainly used to represent sets of objects with respect to their proximities in a low dimensional Euclidean space. Widely used optimization algorithms try to improve the representation via shifting its coordinates in direction of the negative gradient of a corresponding fit function. Depending on the initial configuration, the chosen algorithm and its parameter settings there is a possibility for the algorithm to terminate in a local minimum.

This article describes the combination of an evolutionary model with a non-metric gradient solution method to avoid this problem. Furthermore a simulation study compares the results of the evolutionary approach with one classic solution method.

1 Introduction

Multidimensional scaling is very common in exploratory data analysis. It is mainly used to represent sets of objects with respect to their proximities in a low dimensional Euclidean space to get a better insight into the underlying structure of the data. The low dimensional configuration X of the data is measured with respect to its goodness of fit for example by a non-metric stress function $S(X)$ (see Kruskal (1964a)). Widely used optimization algorithms try to improve $S(X)$ via shifting each configuration's coordinates

in direction of the negative gradient of $S(X)$ (see e.g. Kruskal (1964b), Johnson (1973), Meulman and Verboon (1993), Busing et al. (1997)). Depending on the initial configuration, the chosen algorithm and its parameter settings there is a possibility of the algorithm to terminate in a local stress minimum. Thus, the resulting final configuration may not represent the structure of the data accordingly.

Recent publications try to address this problem by proposing algorithms which use models like e.g. neural networks (in van Wezel et al. (2001)) or simulated annealing (Klock and Buhmann (1997)) to overcome that problem. An often suggested alternative to avoid local minima of the stress function is to use more than one starting configuration (see e.g. Opitz (1980), Webb (1999), Young (1987)). The aim of this study is to explore the capabilites of an evolutionary based algorithm with regard to this problem. Therefore a combination of an evolutionary model using a classic non-metric gradient solution method has been developed and implemented to

- achieve more heterogeneity on the solution search space,
- to enable configurations to exchange informations with crossover techniques and
- to randomly change the algorithm's parameters with mutation

and therefore prevent solutions from sticking to a local minimum.

2 The Gradient Method

We assume n multivariate data objects y_i ($i \in \{1,\ldots,n\}$) have to be represented with respect to their proximities in a low dimensional Euclidean space. Each iteration of the algorithm leads to a configuration

$$X^l = \left(x_{ij}\right)^l_{n,k},$$

j denoting each object's position in $\mathbb{R}^k$, k usually $\in \{1,2,3\}$ and l counting the number of the actual iteration. X^0 will be chosen randomly and the gradient step factor of the 0th

iteration λ_0 will be initiated to a small positive real number. The original dissimilarity matrix has the form

$$D = \left(d_{ij}\right)_{n,n} \in \mathbb{R}_+^{n \times n}$$

. Additionally we assume that

- there are no missing values in D,
- $d_{ii} = 0 \quad \forall i \in \{1, \ldots, n\}$
- (i', j') is more similar to $(i, j) \Leftrightarrow d_{i'j'} < d_{ij}$.

The lth iteration of the algorithm can be described as follows:

1. Calculation of $S(X^l)$ (Kruskal (1964a))
   - Calculation of dissimilarity $\hat{d}(i, j)$ with a L_p norm
   - Monotonic transformation of $\hat{d}(i, j)$ into $\delta(i, j)$ in such a way that

     $$d(i, j) \leq d(i', j') \Rightarrow \delta(i, j) \leq \delta(i', j')$$

   - Calculation of raw Stress function $S^*(X^l)$ and scaling it with maximal Stress

     $$S(X^l) = \frac{S^*(X^l)}{S_{\max}(X^l)} = \frac{\sum\limits_{i<j} \left(\hat{d}(i, j) - \delta(i, j)\right)^2}{\sum\limits_{i<j} \left(\hat{d}(i, j) - \frac{2}{n(n-1)} \sum\limits_{i<j} \hat{d}(i, j)\right)^2}$$

2. Shifting points along the negative gradient of X^l (∇S):

   $$X^{l+1} = X^l - \lambda_l \nabla S|_{X^l}$$

3. $\lambda_{l+1} = r\lambda_l$, with $r \in (0, 1)$

Iterations will be repeated till a termination criterion (e.g. minimal Stress) is fulfilled.

3 The Evolutionary Algorithm

Evolutionary strategies are based on biologically motivated reproducing and selection strategies and proved to be useful for solving a large range of problems which occur with optimization algorithms (see e.g. Nissen (1997)). For the given problem we put s MDS configurations in a population $P = \{I(1), I(2), \ldots, I(s)\}$. This Population contains individuals $I(i) = \{X(i), S(i)\}$ with $X(i)$ a randomly generated start configuration and $S(i) = \{\sigma(i)_1, \ldots, \sigma(i)_n\}$ as a set of normally distributed standard deviations ($\sigma(i)_j \sim N(0, r)$, $r \in \mathbb{R}_+$). We speak of a generation of this population as one iteration step of the algorithm. Each generation passes the following steps:

1. Calculation of the fitness $F(X(i)) = f(S(X(i)))$ of each individual with a monotonic decreasing function $f : \mathbb{R}_+ \to \mathbb{R}_+$ of the stress, e.g. $f(r) = r^{-1}$ or $f(r) = ln^{-1}(r)$

2. Random selection of the mating pool and the parents:

   - Save the individual with the best fitness into the mating pool
   - Select randomly $s-1$ (with $\frac{s}{2} \in \mathbb{N}$) individuals out of P into a mating pool $M = \left\{\widetilde{I}(1), \ldots, \widetilde{I}(s)\right\}$ with selection probabilities

   $$p(I(i)) = \frac{f(I(i))}{\sum_{i=1}^{s} f(I(i))}$$

   - Divide mating pool in randomly selected pairs of parents:

   $$M = \left\{ I(1)_{p_X}, I(1)_{p_Y}, \ldots, I\left(\frac{s}{2}\right)_{p_X}, I\left(\frac{s}{2}\right)_{p_Y} \right\}$$

3. Generation of the **children** through

   - **Direct Crossover**

     Each pair of parents I_{p_X} and I_{p_Y} exchange their configurations X_p and Y_p with a certain probability at randomly chosen crossover point(s) c_i ($0 < c_i < n$). Thus the configurations X_c and Y_c of the children I_{c_X} and I_{c_Y} will be set depending on the strategy type (see figures 1 and 2) to

$$X_c(i) = \begin{cases} X_p(i), \text{ if } i \geq c \\ Y_p(i), \text{ if } i < c \end{cases}, Y_c(i) = \begin{cases} Y_p(i), \text{ if } i \geq c \\ X_p(i), \text{ if } i < c \end{cases} \text{ (Strategy 1), or}$$

$$X_c(i) = \begin{cases} X_p(i), \text{ if } i \neq c \\ Y_p(i), \text{ if } i = c \end{cases}, Y_c(i) = \begin{cases} Y_p(i), \text{ if } i \neq c \\ X_p(i), \text{ if } i = c \end{cases} \text{ (Strategy 2)}$$

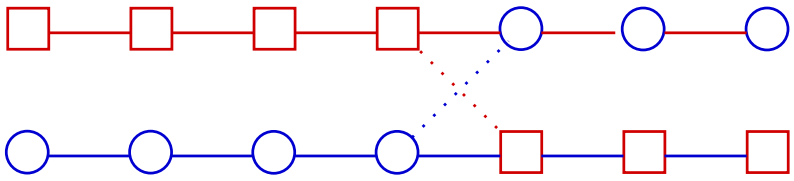

Figure 1: Crossover Strategy 1

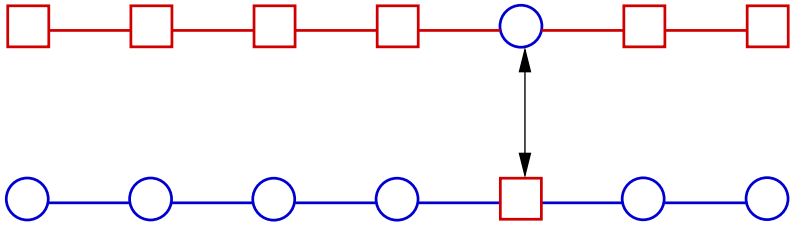

Figure 2: Crossover Strategy 2

- **Intermediate or Discrete Recombination and Mutation**

  A standard deviation $\sigma(i)_j$ ($j \in \{1,...,n\}$) is assigned to each individual's point. Those standard deviations serve as a base for the random mutation process:

  (a) The Children's standard deviations of mutation $(\sigma_c(i_1)_j, \sigma_c(i_2)_j)$ are calculated according to:

  i. An intermediate recombination of the parent's standard deviations $(\sigma_p(i_1)_j, \sigma_p(i_2)_j)$, e.g. $\sigma_c(i_1)_j = \sigma_c(i_2)_j = \frac{1}{2}\left(\sigma_p(i_1)_j + \sigma_p(i_2)_j\right)$ for all $j \in \{1,...,n\}$ or

  ii. A Discrete recombination $\sigma_c(i_1)_j = \sigma_p(i_2)_j$ and $\sigma_c(i_2)_j = \sigma_p(i_1)_j$ for randomly selected $j \in \{1,...,n\}$

  (b) randomly select points to mutate

(c) $\widetilde{X}_c(i)_{jk} = X_c(i)_{jk} + Z(i)_{jk}$ with $Z(i)_{jk} \sim N(0, \sigma(i)_j)$

- **Mutation of Gradient Factor**
  (a) $\widetilde{\lambda}$ is the gradient factor of the fittest parental individual
  (b) Set the gradient factor of each child to $\lambda(i) = \widetilde{\lambda} \cdot b^Z$ with $Z \sim U(-1,1)$ and b a positive real constant
  (c) Set $\widetilde{\widetilde{X}}_c(i) = \widetilde{X}_c(i) - \lambda(i) \cdot \nabla S|_{\widetilde{X}_c(i)}$

Generations will be calculated by starting over at (1) till a termination criterion has been reached. The fittest individual, i.e. the one containing the configuration with the smallest stress, is taken as the stress value assigned to the generation.

4 The Simulation Study

To compare both the evolutionary and the classic approach, sets of randomly generated data objects served as a source for three simulation experiments. The representation space was chosen to be $\mathbb{R}^2$. Each experiment consisted of 500 MDS solution runs and was terminated either by reaching a given minimal stress value or by exceeding a given time limit. Some of the algorithm's parameters have been set to fixed values throughout the simulation runs (see table 1). The parameters controlled by the Monte Carlo Analysis are set by uniformly distributed random variables (see table 2).

The input data sets for each experiment type have been generated randomly as well:

Type A: 30 normally distributed objects y with attributes $y(i) \sim N(0,1)$,

Type B: 6 sharp clusters of 5 objects each with

$$y(i) = c_j(i) + Z, \quad c_j(i) \in \{0,1\}, \quad Z \sim N(0, 0.1)$$

Type C: 5 sharp clusters of 5 objects each and 5 wide spread objects as outliers

$$y(i) = c_j(i) + Z_j \quad \text{with} Z_j \sim \begin{cases} N(0,0.1), \text{ if } j \in \{1,...,5\} \\ N(0,10) \text{ , if } j = 6 \end{cases}$$

	Data Type		
Parameter	**A**	**B**	**C**
Random Mutation Jump width	0.1	0.1	0.1
Crossover Strategy	2	2	2
Recombination Type	discrete	discrete	discrete
Mutation Probability	1	1	1
Maximal Time [sec.]	50	200	50
Initial Gradient Factor λ	0.5	0.5	0.5
Max. Factor for Gradient Change	10	10	10
Minimal Stress	$6 \cdot 10^{-3}$	$6 \cdot 10^{-3}$	$1 \cdot 10^{-4}$

Table 1: Setting of fixed parameters

Parameter	data Type	**min**	**max**	**step**
Population Size	all	2	40	2
Classic or Evolutionary	all	c	e	-
	A	0.2	1	1e-5
$\triangle\lambda$ (Classic)	B	0.7	1	1e-5
	C	0.7	1	1e-5
Crossover Probability	all	1e-3	1	1e-4

Table 2: Monte Carlo Analysis: Uniformly Distributed Parameters

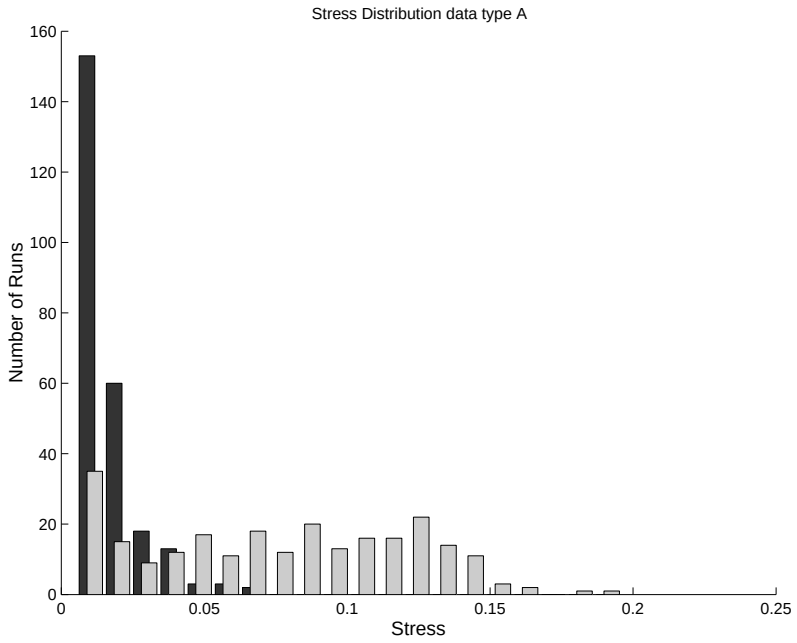

Figure 3: Monte Carlo Simulation Results of Data Type A

($i \in \{1,2,3\}$ denoting dimension, $j \in \{1,...,6\}$ number of cluster and c_j center of cluster)

The dissimilarity matrix D of the generated input data is calculated by aggregating the normed values of each attribute with a L_2 norm:

$$d_{ij} = \sqrt{\sum_{k=1}^{3} \frac{\left(x_{ik} - x_{jk}\right)^2}{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ik} - \bar{x_k}\right)^2}}, \text{ with } \bar{x_k} = \frac{1}{n}\sum_{i=1}^{n} x_{ik}$$

5 Results

Figures 3, 4 and 5 show a histogram of the stress values reached after termination of each run. The results of the classic runs have been collected in the grey bars, the black bars show the results of the evolutionary based runs. The classic results show a distribution with greater variance and larger worst case values in comparison to the evolutionary ones. The worst, best, average and median stress values are summarized in table 3. The worst case runs of the evolutionary approach resulted in final stress values with factor 0.12-0.35 smaller than the runs of the classic algorithm. The resulting configurations of the worst

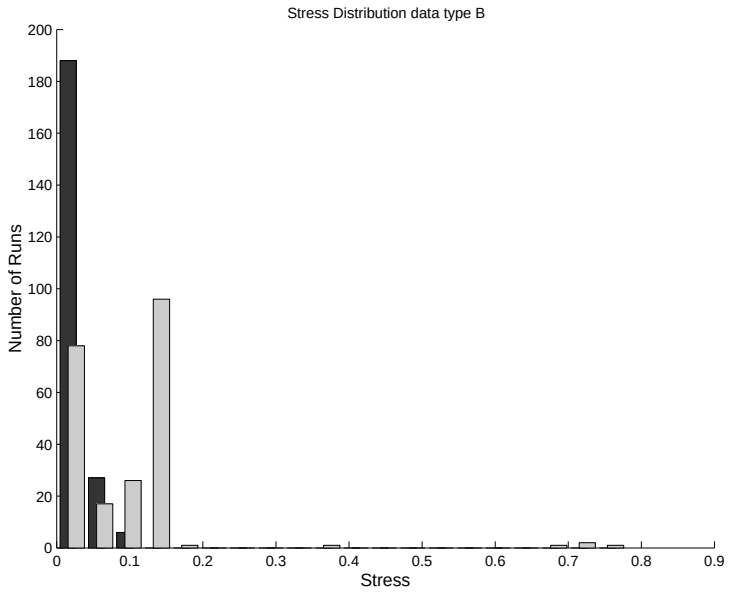

Figure 4: Monte Carlo Simulation Results of Data Type B

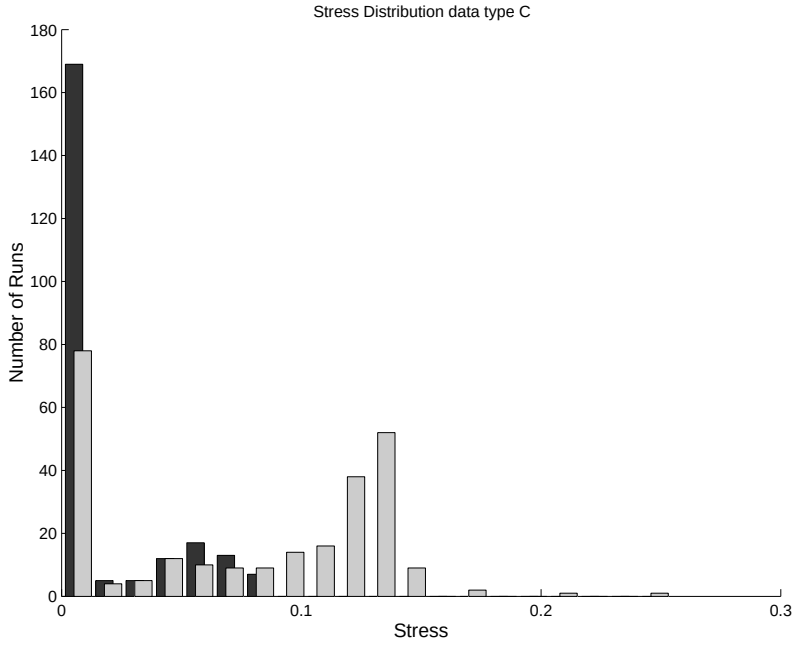

Figure 5: Monte Carlo Simulation Results of Data Type C

case runs for 2 dimensional representations have been plotted for data type **B** in figures 6 and 7 and for data type **C** in figures 8 and 9 (see appendix, pages 14f). The 6 clusters are distinguishable in the evolutionary result only. The corresponding plots of data type **A**[1] have been left out because they are not expected to show a visible structure.

A interpretable dependency of the parameters varied during the Monte-Carlo analysis could not be found. Probabely this is a result of short simulation runs and wide spread parameter ranges.

For graphed details concerning interesting parameter pairs refer to figures 10 to 18 (see appendix, pages 14ff). The filled circles in the plots show the evolutionary runs, the triangles belong to the classic solution attempts.

Thereby figures 10 to 12 show the duration against the reached stress for each simulation run. The abortion criterions of the algorithm explain the distribution of the resulting points. With data type **A** (see figure 10) many runs could reach the minimal stress ($6 \cdot 10^{-3}$) but more runs failed in obtaining it because of the time limit[2] (50 sec.). Data type **B** shows a similar structure with the exception of many runs reaching smaller stress values than specified. The probability for each run to make a larger progress in one algorithmic step was apparently greater with data type **B** than with data type **A**. The outliers with very large stress values in fig. 11 resulted from a unlucky initial representation and the scaling of the gradient scaling factor with a value of 1, so the gradient scaling factor stayed by its initial value and the (even local) minimum could never be reached. In figure 12 the minimal stress value (10^{-4}) could never be reached so each result shows time values around 50 seconds.

In figures 13 to 15 the change factor of the gradient scaling factor has been plotted against the reached stress values for each run. Figure 13 shows that values from around 0.7 to 1 should be chosen to increase the probability to reach smaller stress values.

The effect of the population size, plotted against the reached stress values for the evolutionary runs (figures 16 to 18), could not be detected with the given simulation.

[1](30 normally distributed objects with equal mean and standard deviation)

[2]Time was measured after each run so the evolutionary runs with larger populations reached greater durations

		Data Type		
		A	**B**	**C**
evolutionary	worst	0.070	0.092	0.089
	best	0.0056	0.0019	0.00067
	average	0.015	0.014	0.015
	median	0.0096	0.0042	0.0073
classic	worst	0.20	0.78	0.25
	best	0.0057	0.0020	0.00068
	average	0.076	0.096	0.076
	median	0.076	0.10	0.097

Table 3: Stress value results of simulation runs

6 Drawbacks and Outlook

One drawback of the evolutionary approach is the high computation costs at large number of individuals in one population in comparison to the classic algorithm. If one algorithmic termination criterion is a maximum time, a large population could result in a small number of generations with a final stress not representing the capabilities of the algorithm. So the number of individuals together with the maximal time limit should be chosen carefully. Another disadvantage could lie in the direct crossover of points. Actually it leads to a higher heterogeneity, but as long as the configurations don't have a rotation and translation invariant form an improvement of the configuration with direct crossover should be rather casual.

Therefore the algorithm is planned to be improved and optimized with regards to the following points:

- Optimization of the algorithm:
  - Transform each individual into a rotation and translation invariant form.
  - Set up a measure for similarity of individuals to assign a penalty function to individuals similar to the ones already selected in the mating pool, i.e. change

the fitness of those e.g. to

$$\widetilde{F}(I(i)) = \frac{F(I(i))}{p},$$

with $p \in (1, \infty)$ as a penalty coefficient.

- Individualize the evolutionary changed gradient scaling factor for each point and adapt cross-over routines accordingly.
- Optimize the default values of the algorithm's parameters with a larger simulation study.

- Compare results with other actual algorithmic approaches (e.g. as in van Wezel et al. (2001), Busing et al. (1997) or in Klock and Buhmann (1997))
- Include software module in a Java and R based data mining system under work

7 Conclusion

Evolutionary algorithms seem to be promising for solving non-metric multidimensional scaling problems. With the first version of the algorithm presented with this study the quality of the resulting configurations with respect to smaller stress values could be improved significantly in comparison to one classic algorithm. A simulation study shows that the worst case runs of the evolutionary approach result in final stress values for the given sets of simulated data which are by factor 0.12-0.35 smaller than the classic algorithm's runs.

List of Tables

References

Busing, F., Commandeur, J., and Heiser, W. (1997). PROXSCAL: A multidimensional scaling program for individual differences scaling with contraints. *Softstat*.

Johnson, R. M. (1973). Pairwise nonmetric multidimensional scaling. *Psychometrika, 38*, pages 11–18.

Klock, H. and Buhmann, J. M. (1997). Multidimensional scaling by deterministic annealing. In Pelillo, M. and Hancock, E. R., editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 245–260.

Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika, 29*, pages 1–27.

Kruskal, J. B. (1964b). Nonmetric multidimensional scaling: A numerical method. *Psychometrika, 29*, pages 115–129.

Meulman, J. and Verboon, P. (1993). Points of view analysis revisited: fitting multidimensional structures to optimal distance components with cluster restrictions on the variables. *Psychometrika, 58*, pages 7–35.

Nissen, V. (1997). *Einführung in evolutionäre Algorithmen*. Vieweg, Braunschweig u.a.

Opitz, O. (1980). *Numerische Taxonomie*. Gustav Fischer Verlag, Stuttgart, New York.

van Wezel, M. C., Kosters, W. A., van der Putten, P., and Kok, J. N. (2001). Nonmetric multidimensional scaling with neural networks. *Lecture Notes in Computer Science*, 2189:pp. 245–260.

Webb, A. (1999). *Statistical Pattern Recognition*. Oxford University Press, Inc, New York.

Young, F. W. (1987). *Multidimensional Scaling - history, theory and applications*. Lawrence Erlbaum Associates, Inc., Publishers.

Appendix: Graphics

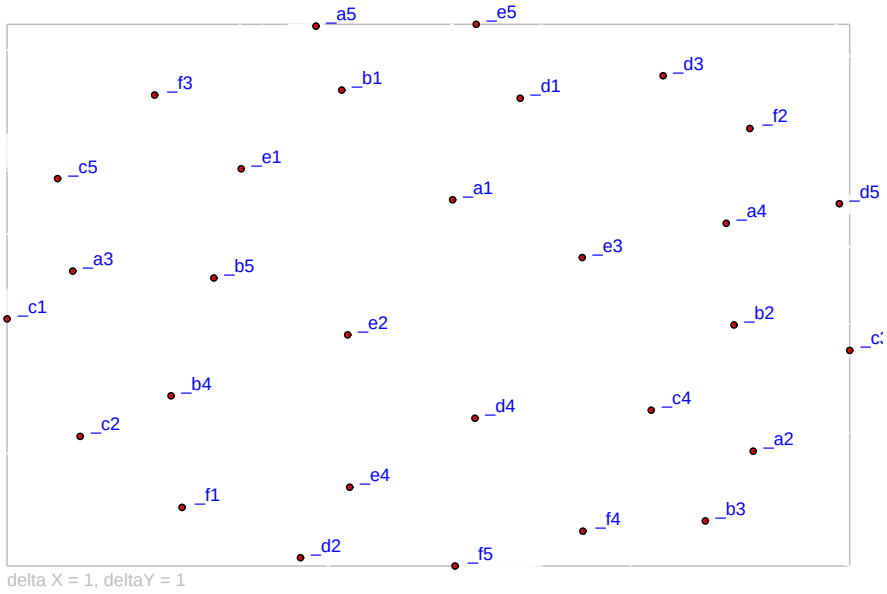

Figure 6: worst configuration of classic runs with data type B

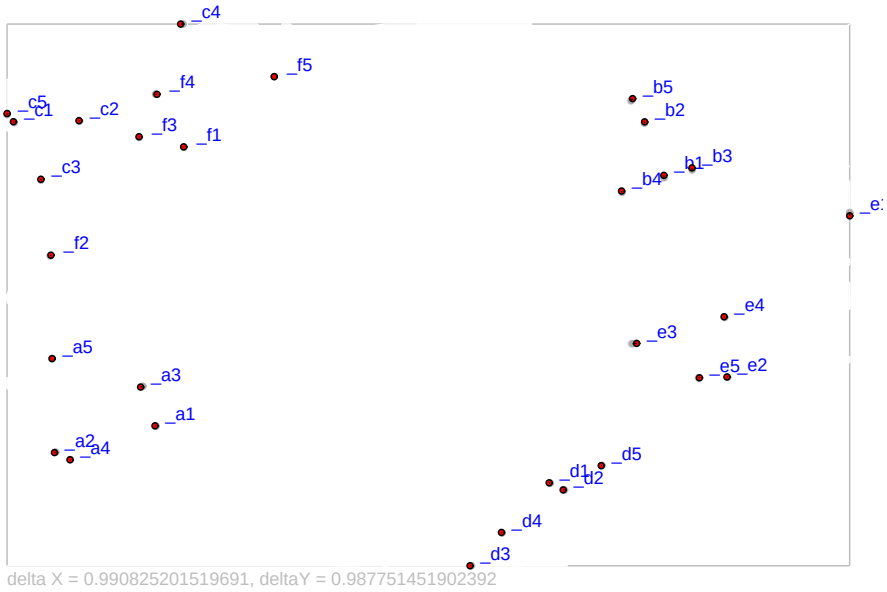

Figure 7: worst configuration of evolutionary runs with data type B

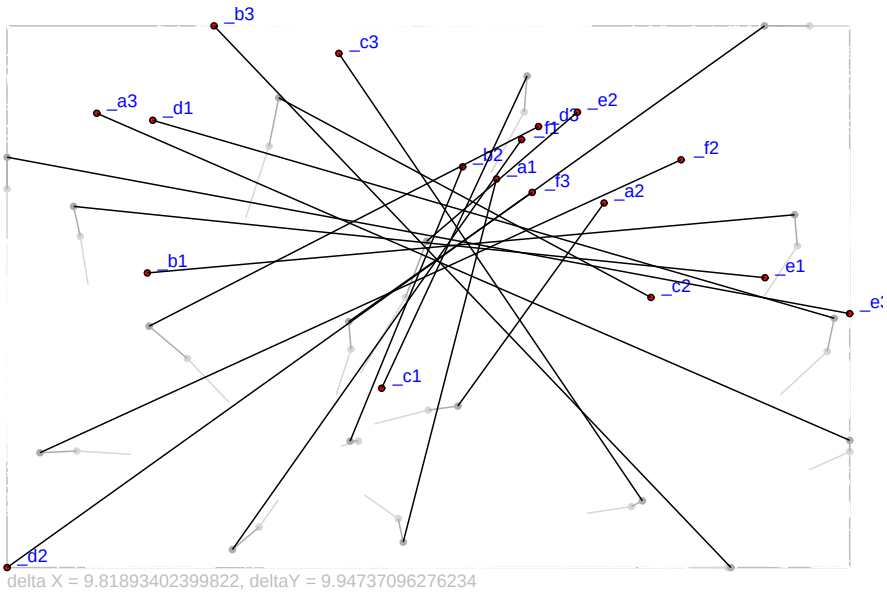

Figure 8: worst configuration of classic runs with data Type C

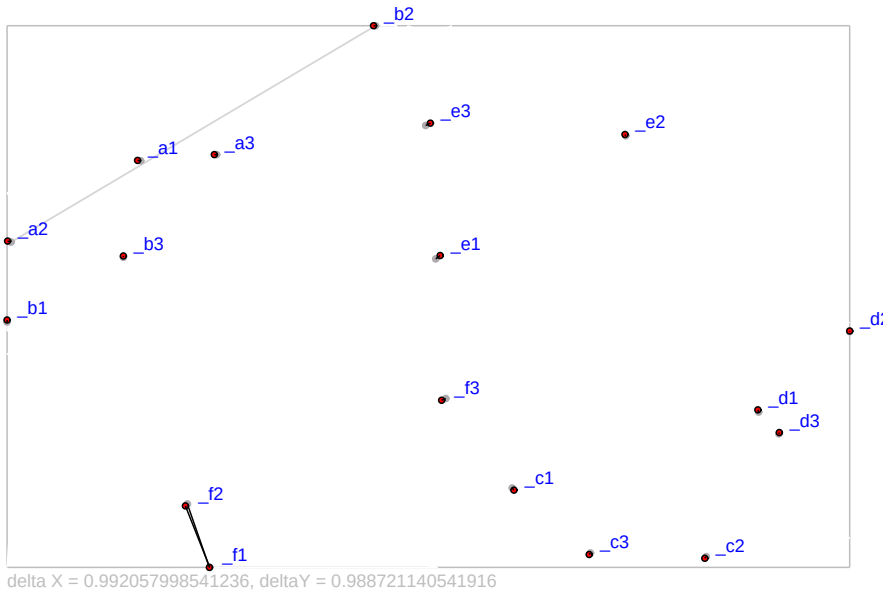

Figure 9: worst configuration of evolutionary runs with data type C

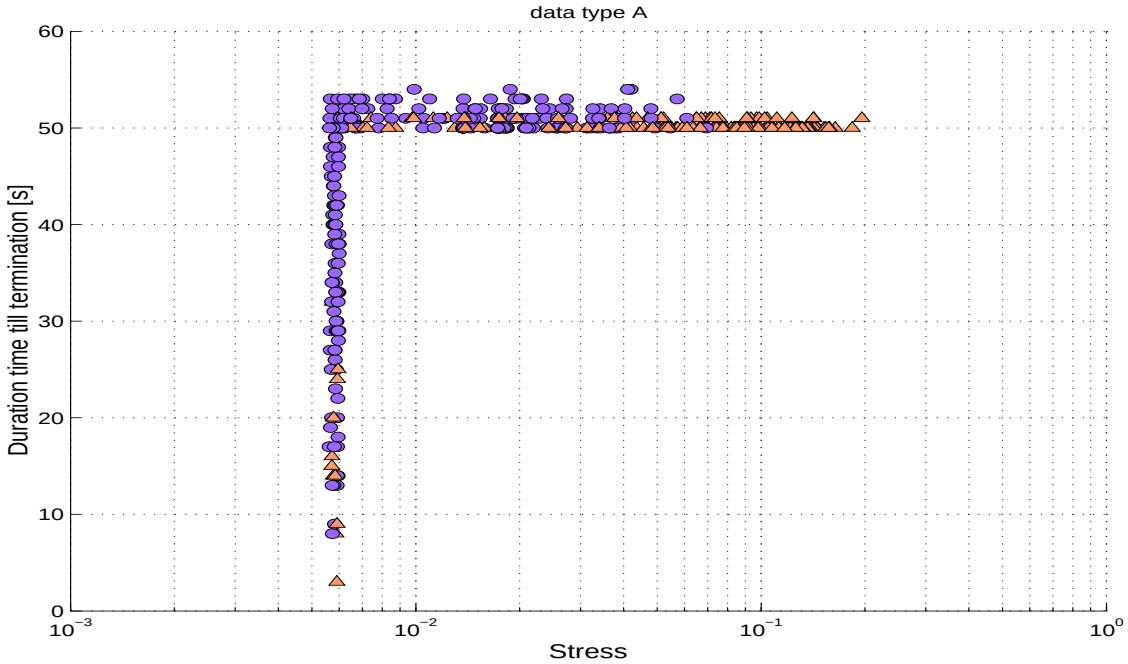

Figure 10: duration against stress for data type A

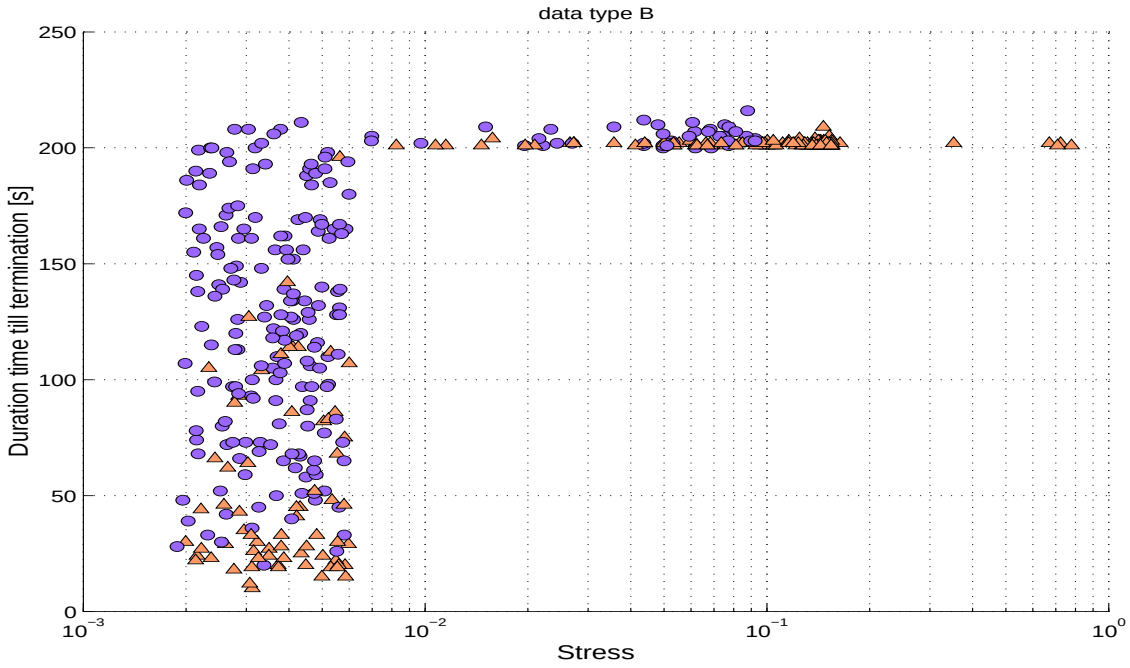

Figure 11: duration against stress for data type B

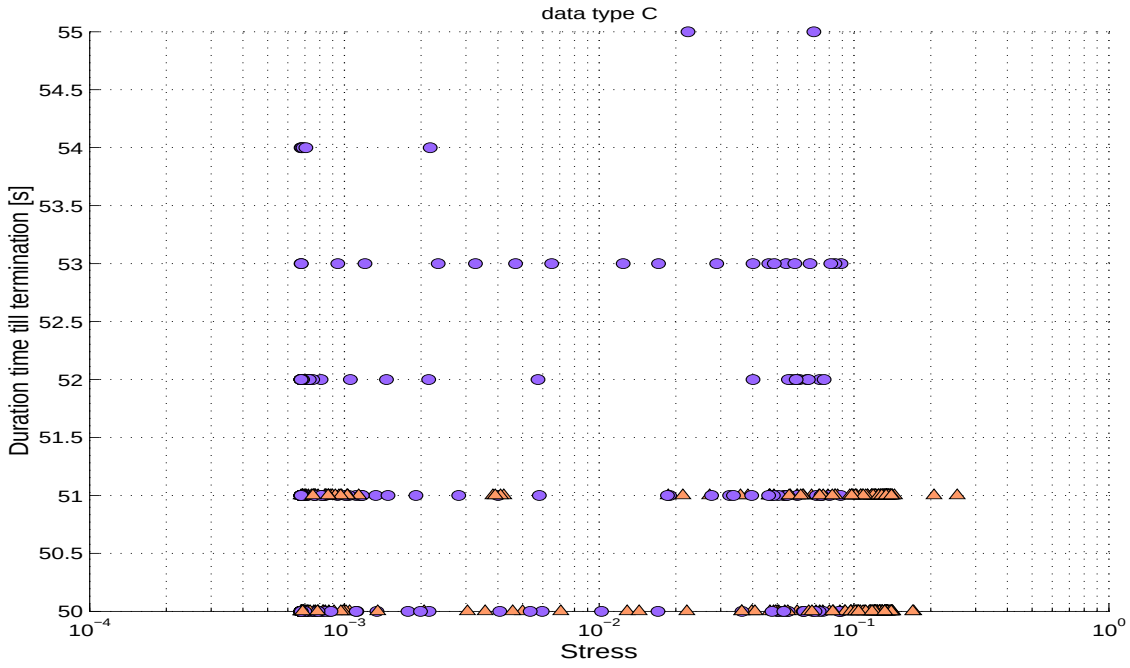

Figure 12: duration against stress for data type C

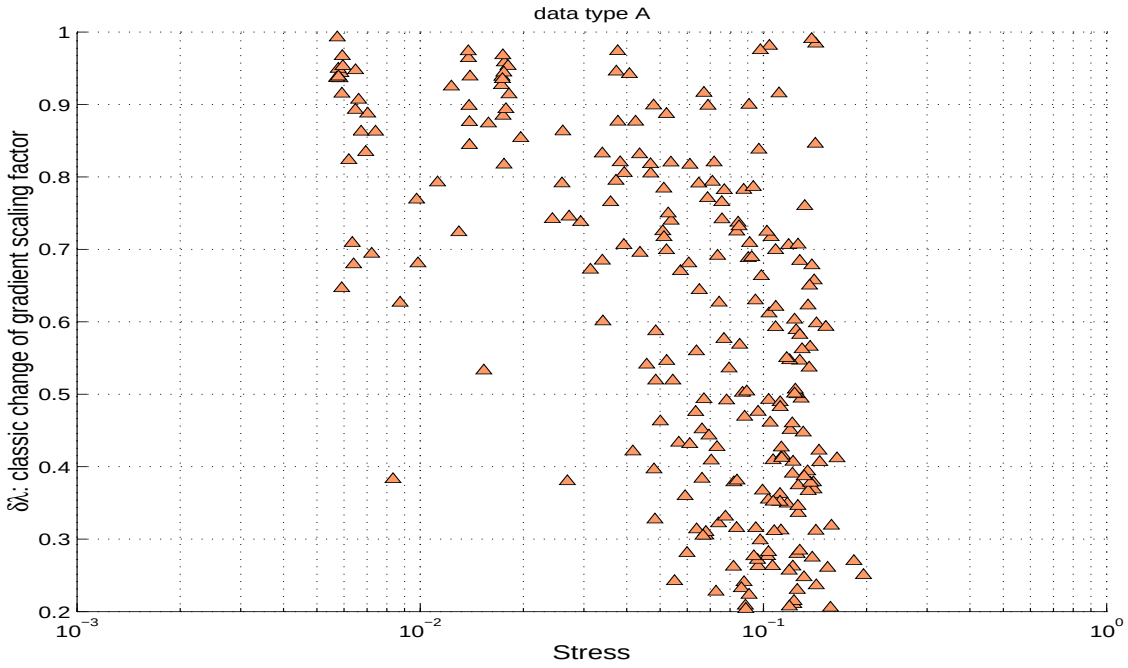

Figure 13: λ against stress for data type A

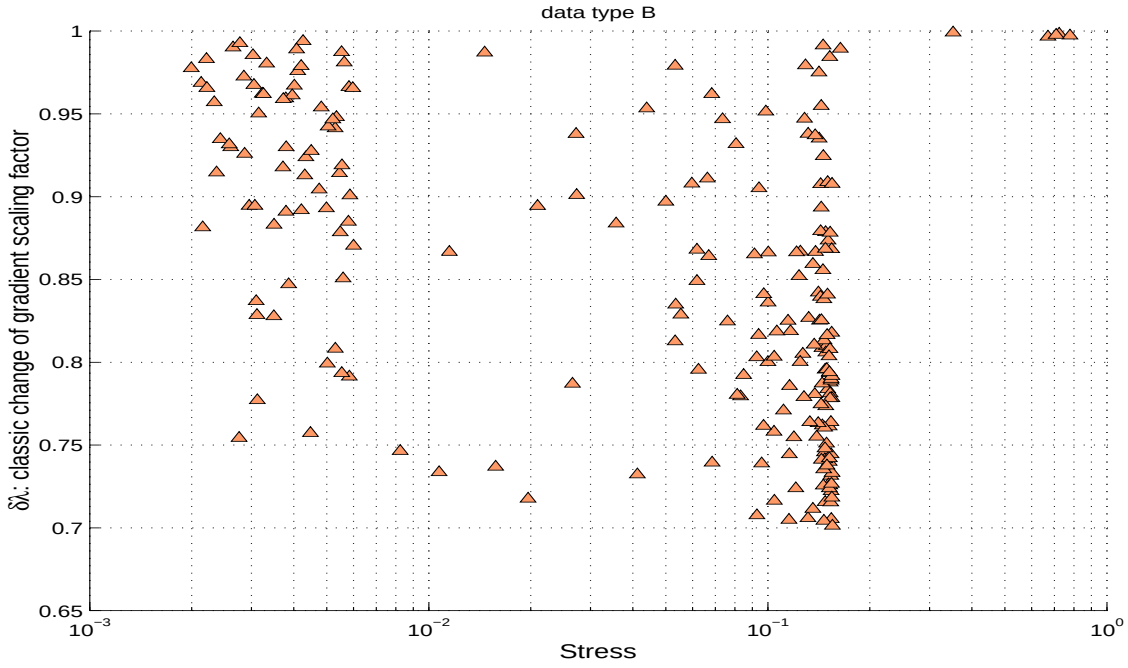

Figure 14: λ Against Stress for data type B

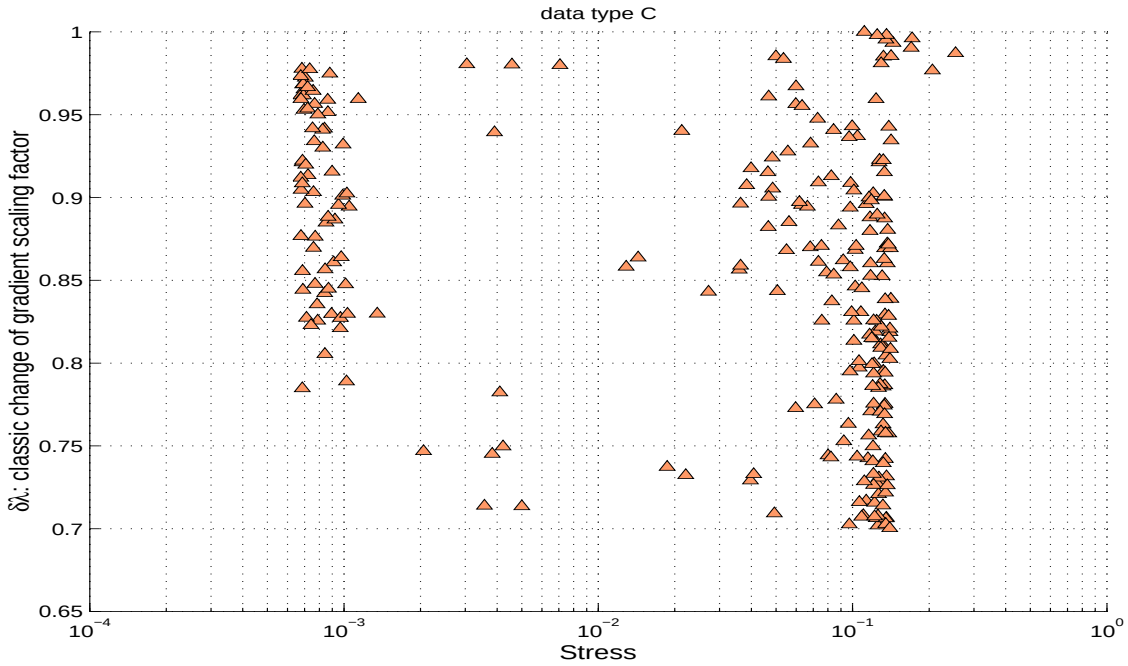

Figure 15: λ against stress for data type C

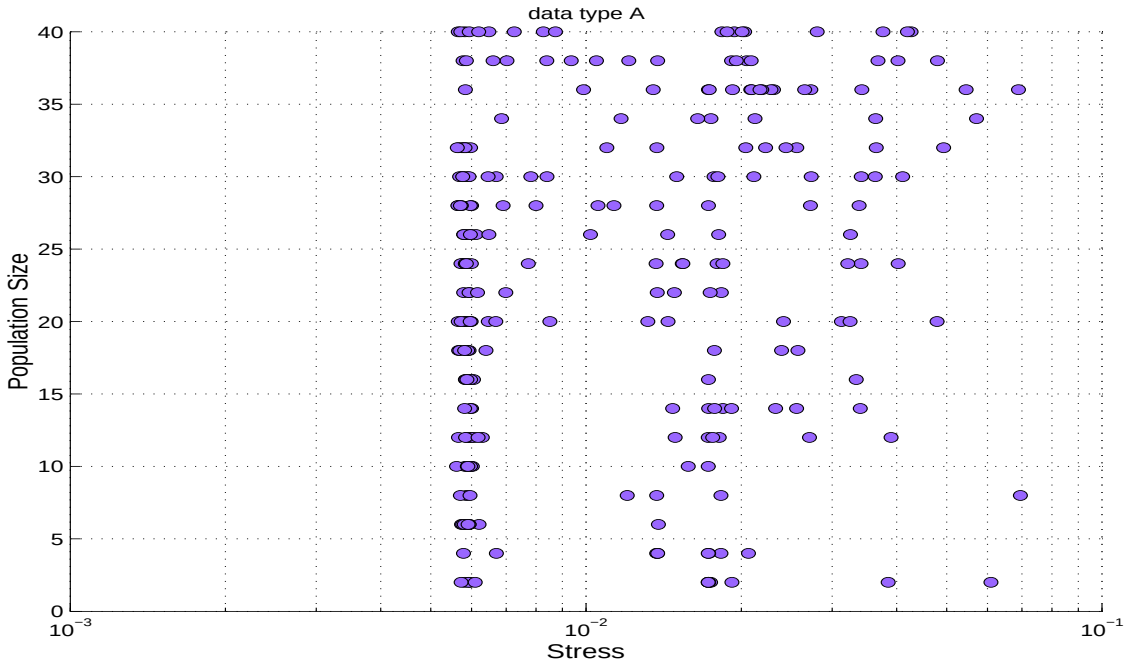

Figure 16: population size against final stress for data type A

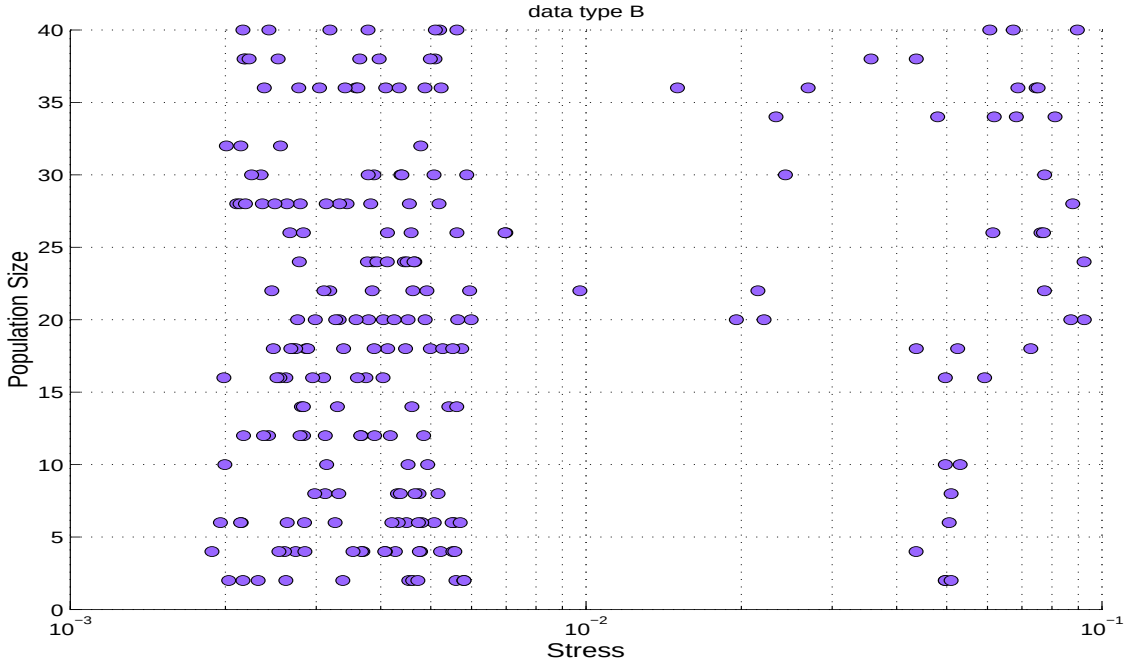

Figure 17: population size against final stress for data type B

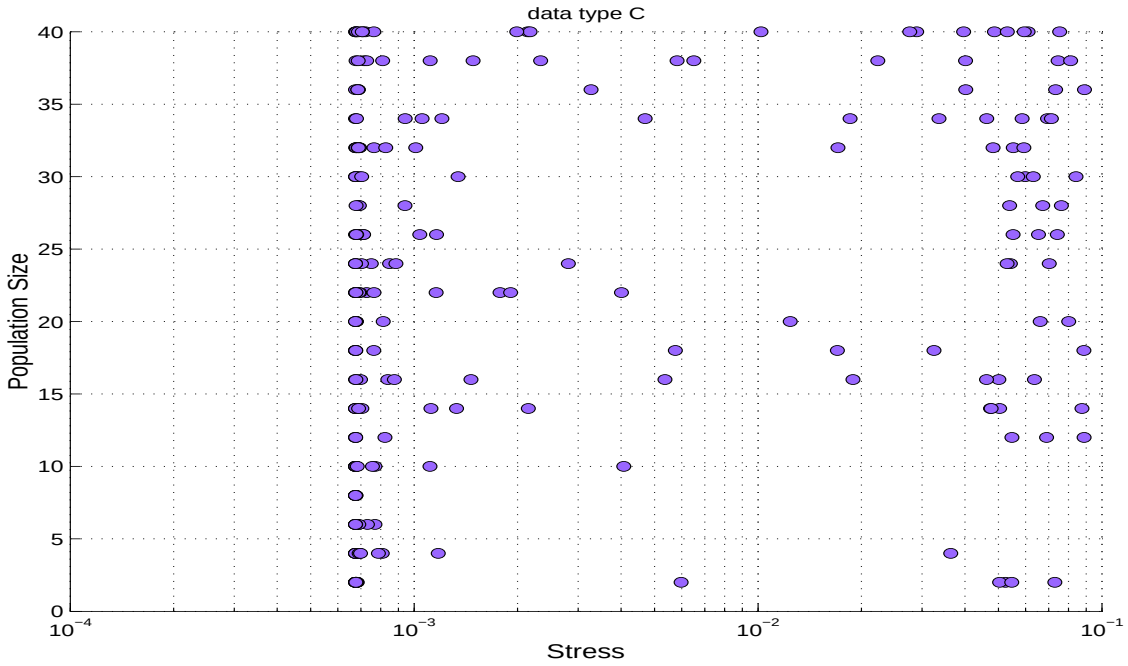

Figure 18: population size against final stress for data type C