Comparing Evolutionary Algorithms on the Problem of Network Inference

Christian Spieth
Centre for Bioinformatics
Tübingen (ZBIT)
72076 Tübingen, Germany
spieth@informatik.uni-tuebingen.de

Rene Worzischek
Centre for Bioinformatics
Tübingen (ZBIT)
72076 Tübingen, Germany
renewor@web.de

Felix Streichert
Centre for Bioinformatics
Tübingen (ZBIT)
72076 Tübingen, Germany
streiche@informatik.uni-tuebingen.de

ABSTRACT

In this paper, we address the problem of finding gene regulatory networks from experimental DNA microarray data. We focus on the evaluation of the performance of different evolutionary algorithms on the inference problem. These algorithms are used to evolve an underlying quantitative mathematical model. The dynamics of the regulatory system are modeled with two commonly used approaches, namely linear weight matrices and S-systems and a novel formulation, namely H-systems. Due to the complexity of the inference problem, some researchers suggested evolutionary algorithms for this purpose. However, in many publications only one algorithm is used without any comparison to other optimization methods. Thus, we introduce a framework to systematically apply evolutionary algorithms and different types of mutation and crossover operators to the inference problem for further comparative analysis.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Miscellaneous*; J.3 [**Computer Applications**]: LIFE AND MEDICAL SCIENCES—*Biology and genetics*

General Terms

ALGORITHMS, PERFORMANCE

Keywords

Evolutionary Computation, Inference, Systems Biology

1. INTRODUCTION

Systems biology has become one of the major research areas in biology over the past few years. Due to tremendous progress in experimental methods like DNA microarrays, several thousand expression levels of genes in an organism can be measured in parallel under specific environmental conditions. This enables researchers to examine intracellular processes on a systemic level. The inference of gene regulatory networks from experimental data is one of the main unsolved problems in the post-genomic area. A gene regulatory network (GRN) is an abstract model representing dependencies between genes using a directed graph. In this graph, each node is a gene or component of the regulatory system and each edge represents a regulatory impact from one component to the other (e.g. activation or suppression of the transcription and translation of the dependent gene).

Several publications addressing the problem of inferring gene regulatory networks can be found in the literature. De Jong gives a good overview about related work in [1]. A major part of the work done in this field is using deterministic mathematical models to simulate regulatory networks. One kind of those deterministic models are linear models like the weighted matrix model [16, 17]. These models have only a small number of system parameters compared to S-systems but are often not flexible enough to model biological systems in detail, since they model the dependencies linearly. S-systems, on the other hand, model dynamic systems in a nonlinear manner. They consist of a set of differential equations describing the changes in expression over time. However, they show a significant higher number of system parameters. S-systems have been recently examined in [5, 6]. Most applications of deterministic models use evolutionary algorithms (EA) to determine the correct parameters of the mathematical model. EAs have proven to be successful in finding parameters of mathematical models representing GRNs.

So far, the authors commonly describe only the algorithm of their choice on the problem of identifying the unknown parameters of the model. Some publications show the results of the proposed method only in comparison to one standard algorithm, thus lacking the possibility of impartially evaluating the performance of other approaches. Therefore, we designed a framework to systematically compare optimization algorithms on a set of artificial benchmark problems, which is accessible through the JCell project website[1] [13].

The remainder of this paper is structured as follows. Section 2 describes the mathematical models used in this publication. Section 3 gives the results of eight commonly used algorithms on the inference problem and section 4 shows the performance of three fine-tuned methods. The conclusions and an outlook are given in section 5.

[1]http://www.jcell.de

2. MATHEMATICAL MODELING

The genetic dependencies of a cell can be abstracted by a directed graph with N nodes representing N genes. Each gene g_i produces a certain amount of mRNA x_i when expressed and changes the concentration of the mRNA level over time: $\vec{x}(t+1) = h(\vec{x}(t))\,,\, \vec{x}(t) = (x_1, \cdots, x_n)$. Here, function h represents the changes of expression levels from one state to the next. To model this function, several approaches can be found in the literature. We decided to use the two most popular deterministic models, namely linear weight matrices, H-systems, and S-systems. These models are described in detail in the following sections.

2.1 Linear Weight Matrices

Linear weight matrices (WM) have been originally introduced in [16]. In this approach, the regulative interactions between the genes are represented by a weight matrix, $\mathcal{W}$, where each row of $\mathcal{W}$ represents all the regulatory inputs for a specific gene. The regulatory effect of gene g_j on gene g_i at time t is simply the expression level of g_j multiplied by its regulatory influence on g_i, w_{ij}. The total regulatory input to g_i is derived by summing across all the genes in the system and in the following referred to as $r_i(t)$:

$$r_i(t) = \sum_j w_{ij} x_j(t) \qquad (1)$$

Here, a positive value for w_{ij} indicates that gene g_j is stimulating the expression of gene g_i. Similarly, a negative value indicates repression, while a value of zero indicates that gene g_j does not influence the transcription of gene g_i. By modeling regulatory interactions with a weight matrix, we can use mathematical matrix approaches found in the field of neural networks for subsequent analyses of the resultant models. With the regulatory state of each gene, we are now able to model the response of each gene to the given input. The impact of $r_i(t)$ on gene g_i is calculated using a so called "squashing" function (Eqn. 2).

$$x_i(t+1) = \frac{m_i}{1 + e^{-(\alpha_i r_i(t) + \beta_i)}} \qquad (2)$$

where $r_i(t)$ is the mentioned regulatory state of gene g_i, and α_i and β_i are gene specific constants that define the shape of the squashing function for gene g_i. The resulting expression level is only a relative value between 0 and 1, with 0 representing complete repression and 1 representing maximal expression. Thus, these relative levels have to be converted into the real expression space. In addition, the genes can have different levels of maximal expression. Hence, we multiply the calculated relative gene expression level x_i by the maximal expression level for each gene m_i, to get the final expression level for g_i $x_i(t+1)$ as shown in equation (2).

2.2 S-systems

Another, more flexible type of model, are S-systems (SS). They employ a general formalism, which allow for capturing the nonlinearity and general dynamics of the gene regulation. S-systems are a type of power-law formalism, which have been suggested by [11] and can be described by a set of nonlinear differential equations:

$$\frac{dx_i(t)}{dt} = \alpha_i \prod_{j=1}^{N} x_j(t)^{\mathcal{G}_{i,j}} - \beta_i \prod_{j=1}^{N} x_j(t)^{\mathcal{H}_{i,j}} \qquad (3)$$

where $\mathcal{G}_{i,j}$ and $\mathcal{H}_{i,j}$ are kinetic exponents, α_i and β_i are positive rate constants and N is the number of genes in the system. The equations in (3) can be seen as divided into two components: an excitatory and an inhibitory component. The kinetic exponents $\mathcal{G}_{i,j}$ and $\mathcal{H}_{i,j}$ determine the structure of the regulatory network. In the case $\mathcal{G}_{i,j} > 0$, gene g_j induces the synthesis of gene g_i. If $\mathcal{G}_{i,j} < 0$, gene g_j inhibits the synthesis of gene g_i. Analogously, a positive (negative) value of $\mathcal{H}_{i,j}$ indicates that gene g_j induces (suppresses) the degradation of the mRNA level of gene g_i.

2.3 H-systems

H-systems are another type of a set of parameterized differential equations. They originate from the idea to enhance weight matrices with an additional term to ensures nonlinearity in the model. They have been suggested by Hadeler and Spieth [3] and have the form of:

$$\frac{dx_i(t)}{dt} = c_i + \sum_k b_{ik} x_k(t) + x_i(t) \sum_k a_{ik} x_k(t) \qquad (4)$$

Although H-systems have the same order of complexity $O(N^2)$ as S-systems, they have a significant advantage. For S-systems, the equilibrium positions are too fixed, which becomes obvious even for the one-dimensional case $\dot{x} = x^{\alpha} - x^{\beta}$. Furthermore, H-systems can be clearly motivated by dividing the equations into a constant rate and a linear term as in case of the weight matrices and extending this with a nonlinear term.

3. STRAIGHT-FORWARD APPROACH

The most straight-forward approach to the problem of mathematically modelling dynamic systems by means of evolutionary algorithms is to choose an appropriate type of model and a suited EA together with the fitness functions described below. Thus, this section gives an overview of the performance of classic algorithms in combination with the different models on the problem of identifying unknown, nonlinear dynamic systems.

Several classic optimization algorithms, which represent the basic algorithms for parameter optimization, were used to fit a model to the given experimental data:

- Monte-Carlo search (MC),
- (multi-start) hill climber (MS-HC),
- binary genetic algorithm (binGA),
- real-valued genetic algorithm (realGA),
- standard evolution strategy (stdES),
- evolution strategy with CMA mutation (cmaES),
- differential evolution (DE), and
- particle swarm optimization (PSO).

All algorithms applied fitness function (5) to determine the quality of a candidate solution and all use the parameter settings that were described as the default parameters by the authors of the corresponding method if not otherwise stated.

3.1 Fitness

For evaluating the quality or fitness of the inferred models, i.e. the similarity of the time dynamics between the experimental and the simulated data, the following equation can be used, referred to as the relative squared error or relative standard error (RSE):

$$f_{RSE} = \sum_{i=1}^{N} \sum_{k=1}^{T} \left\{ \frac{\hat{x}(t_k)_i - x(t_k)_i}{x(t_k)_i} \right\}^2 \tag{5}$$

Here, N is the total number of components of the system, T is the number of sampling points taken from the experimental time series and $\hat{x}$ and x distinguish between estimated data of the simulated model and data sampled in the experiment. This is the most straight-forward way of comparing the system's output to the output of the simulated model.

In the current implementation, a Runge-Kutta method of fourth order is used to integrate the differential equation systems to simulate the mathematical model. The overall optimization problem is then to minimize the fitness values of objective function f_{RSE}. This fitness function has already been used by several publications on this problem and was thus selected to evaluate the model quality.

3.2 Search Space Properties

To better understand the properties of the search space, the benchmark problems were first examined by two simple optimization algorithms. As a first method, the performance of a straight-forward Monte-Carlo search (**MC**) was examined on the inference problem. Monte-Carlo methods are the most simple, random methods for parameter optimization and can be used to evaluate the performance of randomly choosing solutions from the solution space. The second algorithm to explore the search was a standard hill climbing method (**HC**). Hill climbers represent a rather primitive technique as well, but they are useful in simple and low dimensional uni-modal solution spaces. The HC algorithm of this comparison was implemented with a real-value variable encoding and a fixed mutation step size of 0.2.

Further on, the hill climber was evaluated in various different multi-start environments: 1, 10, 25, 50, 100, and 250. To gain a sound statistic, 20 multi-runs were performed for both algorithm types, i.e. each algorithm was repeated 20 times independently to obtain results that are statistically significant. Further on, each algorithm terminated after 100,000 fitness evaluations and the best-of-run solutions were averaged over the multi-runs, if not mentioned otherwise.

Figure 1 gives the results of the Monte-Carlo search and the different multi-start hill climbing methods for the test cases with increasing dimensions (5 and 10 components). The figures in the left columns give the fitness courses of each algorithm including the courses of the random Monte-Carlo search averaged over the 20 multi-runs. The column on the right shows the mean fitness values and the corresponding standard deviations of the six multi-start environments also averaged over the 20 multi-runs. Here, the MC results are not given to focus on the performance of the multi-start hill climber.

The HC algorithms perform very similar on the benchmark systems. As can be seen from the fitness courses, the trend for each algorithm is the same for all test systems. The multi-start hill climber with only a small number of multi-starts find the worst solutions in comparison to the other multi-start algorithms. However, those environments with the highest number of multi-starts drop in their performance. Beside the Monte-Carlo search, each hill climber was able to find very good solutions with respect to the fitness function on the five-dimensional test case. All algorithms suffer from the increase in the dimensionality of the benchmark systems and therefore, the quality of their results decreased in the small- and medium-sized benchmarks. The 25-HC was able to find satisfying results in the five-dimensional example, whereas all algorithms fail to find good solutions in the medium-sized benchmark ($N = 10$ components).

The experiments of the multi-start hill climber clearly show that the search space is highly multi-modal. The algorithms with increased numbers of allowed multi-starts are yielding increased levels of model quality, in particular with the best performance of the 50-start hill climber in the two-dimensional example and the 25-start HC in the others. Further increasing of the multi-starts implies a decrease of the quality. This can be explained by the properties of multi-start environments: on the one hand, the simple one-start HC converges prematurely to local optima, while the multi-start algorithms are better able to cope with this issue. On the other hand, the higher the number of multi-starts is, the lower is the number of evaluations per hill climbing run, i.e. is the overall number of fitness evaluations divided by the number of concurrent multi-starts. Thus, the high multi-start HCs are not able to fully converge to an optimum suffering from the lack of evaluations.

3.3 Classical Optimization Algorithms

To further investigate the inference problem, several standard evolutionary algorithms were used to find suitable model parameters. As already mentioned, six EAs were selected to evolve the kinetic parameters of the benchmark systems. The settings of the algorithms are described below.

The standard evolution strategy (**stdES**) with global mutation ($p_m = 0.8$), discrete one-point crossover ($p_c = 0.2$) used a ($\mu = 5, \lambda = 25$) population scheme. The more sophisticated ($\mu = 5, \lambda = 25$)-ES with covariance matrix adaptation (**cmaES**) evolved solutions only relying on mutation ($p_m = 1.0$) without crossover. The binary genetic algorithm (**binGA**) used one-point mutation ($p_m = 0.1$) and one-point crossover ($p_c = 0.7$), the real-valued GA (**realGA**) global mutation ($p_m = 0.1$) and UNDX crossover ($p_c = 0.8$). Both worked on a population of 250 individuals and selected from them with tournament selection with a group size of eight individuals. For the differential evolution (**DE**), the extended method was used with a population size of 250 individuals, $F = 0.8$, $\lambda_{DE} = 0.4$, and $p_c = 0.4$. And finally, the particle swarm optimizer (**PSO**) optimized the model parameters with 250 particles using a star topology of range 2, $\phi_1 = \phi_2 = 0.6$, an inertia weight $w = 0.8$, and an initial velocity vector of $\vec{v}^i = 0.2\,\forall\, i$.

Beside these settings, the evolution strategies used an extension, where the initial population is significantly larger than the standard λ-sized population. In the current implementation, 250 individuals were created to form the initial population to be better comparable to the other algorithm with much larger populations. This extension is crucial, because it increases the chance of finding stable S-systems in the beginning. The evolution strategies then decrease the size of the population by the standard best selection method

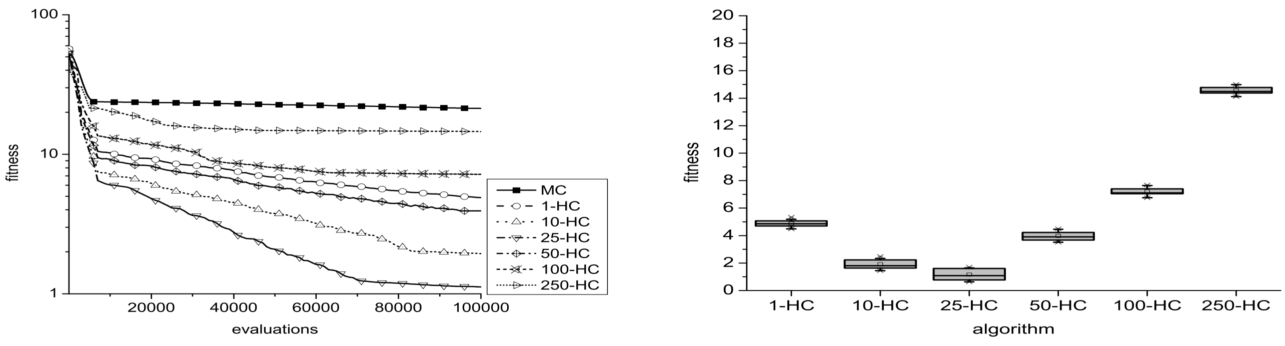


Figure 1: Performance of a standard Monte-Carlo search (MC) and the different multi-start hill climbing methods (x-HC). Given are the fitness courses of the five–dimensional systems.

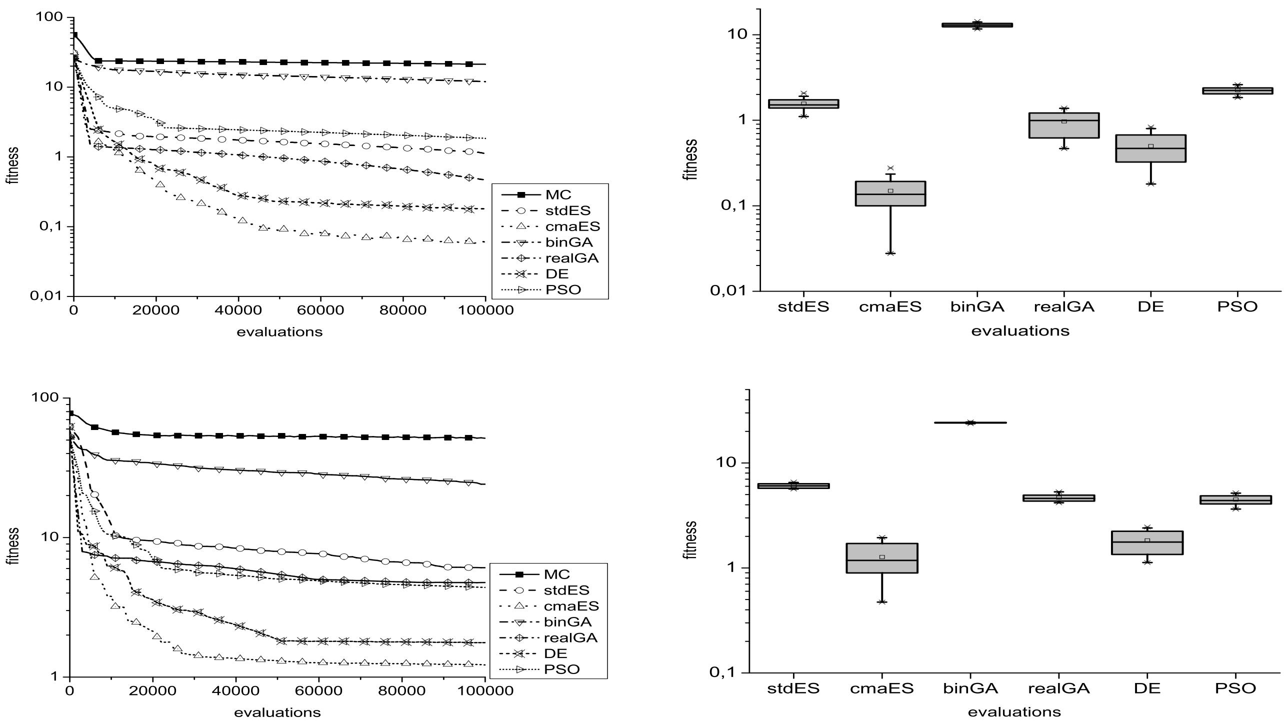


Figure 2: Performance of standard evolutionary algorithms: evolution strategy with global mutation (stdES) and covariance matrix adaptation (cmaES), genetic algorithm with binary (binGA) and real-value (realGA) representation, differential evolution (DE), and particle swarm optimization (PSO). Given are the fitness courses of the five- (upper row) and ten-dimensional systems.

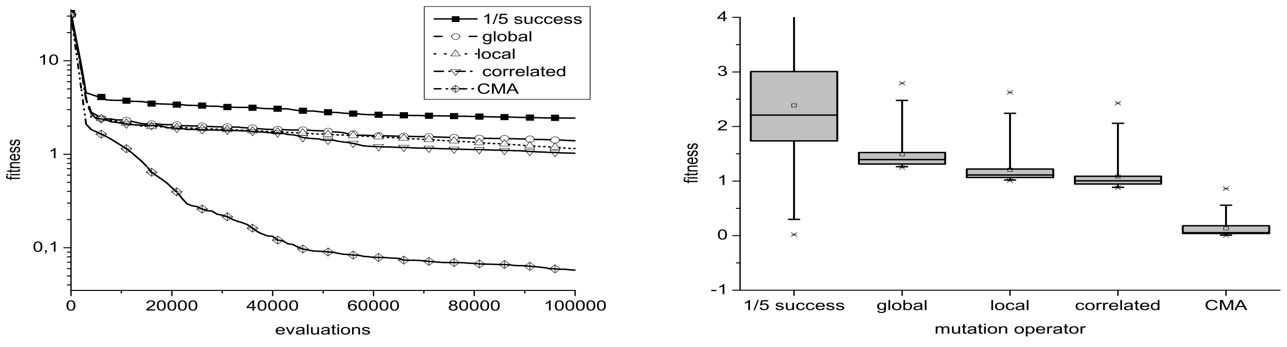


Figure 3: Impact of the mutation operators. Shown is the performance of an ES on the inference of the five-dimensional benchmark systems averaged over 20 multi-runs.

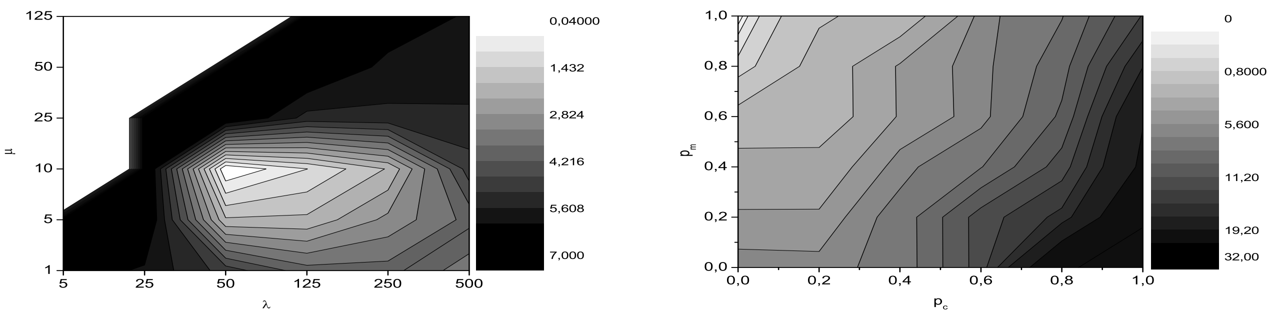


Figure 4: Impact of the μ-λ ratio and the optimal ratio of p_m and p_c for the best performing mutation type. Shown is the performance of an ES on the inference of the five-dimensional benchmark systems.

and continue with the number of individuals given above.

Figure 2 gives the results of the different evolutionary algorithms for the test cases with increasing dimensions. The figures in the left columns give the fitness courses of each algorithm averaged over the 20 multi-runs. The column on the right shows the mean best fitness values and the corresponding standard deviations of the six EAs. The standard ES was not able to find good solutions in the test cases. However, the ES using CMA performed best in each benchmark reflecting the potential of evolution strategies on real-value optimization problems. The binary GA performed hardly better than the random Monte-Carlo search especially in the higher dimensional examples. Whereas the real-valued GA achieved much better results and found good models more reliable. Furthermore, the binGA started with worse fitness values than the realGA, due to the binary representation in which high values are more likely because of the uniform random initialization of high and low bits. The most competitive alternative EA approach to the cmaES was differential evolution, performing second best in all test cases. And finally, the PSO implementation performed rather well but also seemed to converge prematurely.

4. OPTIMIZED ALGORITHMS

The results of the previous section showed that standard EAs are able to find good solutions with respect to the given fitness function. Especially, the ES, the real-valued GA, and the DE performed well and reliably found good solutions. These three EA approaches were selected to be the reference algorithms for the next sections. For this, the algorithm settings are tuned in the following to represent the best suited standard algorithms. The reference algorithms are tuned using only the five-dimensional systems (see JCellDB for details). For this purpose, the different setups of algorithm parameters were evaluated on the inference of the five-dimensional data sets. As in the test cases before, the experiments were repeated 20 times for each setup to gain sound statistics and the resulting fitness values were calculated from the averaged best-of-run values.

4.1 Optimized Evolution Strategy

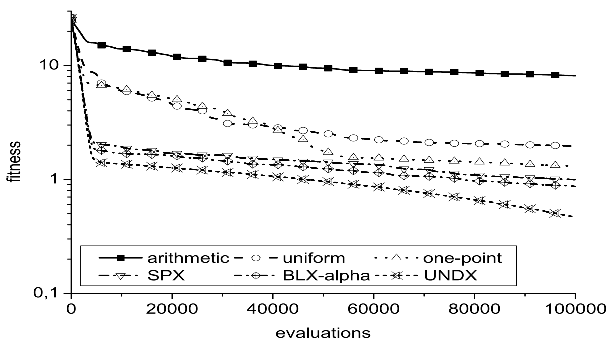
The first reference algorithm to be tuned was an evolution strategy. The results of the preceding section indicate that an ES with an appropriate mutation operator is well suited for the inference problem. Therefore, the impact of the population size, several mutation operators, and the optimal ratio of p_m and p_c for the best performing mutation type were evaluated to find an optimal set of algorithm settings for a reference evolution strategy.

Mutation Operators The results of the plain standard algorithms suggested a strong impact of the type of mutation on the performance of the ES. Thus, five mutation operators were compared to find an operator for the optimizer that results in the best solutions: 1/5 success rule [10], global mutation and local mutation [12], correlated mutation, and covariance matrix adaptation (CMA) [4].

Further on, crossover was omitted for all test cases to study only the impact of the mutation. The graphs of figure 3 give the results of the inference runs averaged over 20 multi-runs. All operators failed to converge to good solutions, only CMA was able to improve the fitness significantly. This suggests that the decision variables of the problem are correlated. And although correlated mutation does not perform significantly better than the other operators, more sophisticated operators that make use of correlation analysis seem better suited for inferring regulatory systems.

Mutation versus Crossover For the preceding evaluation of the mutation operators, no crossover was used. Therefore, additional experiments were performed to study the optimal ratio of mutation and crossover probability, namely p_m and p_c, respectively. This was done in a grid search evaluating different combinations of p_m and p_c for the CMA mutation operator and discrete one-point crossover. Figure 4 shows the results of the optimization runs and clearly, the crossover event disturbs the self-adaption of the mutation operator. The best setting was reached for a mutation rate of $p_m = 1$ and no crossover at all.

Population Size To examine the influence of the population size of the evolution strategy, different combinations of μ and λ were used to infer the inference problem given by the five-dimensional data sets. In particular, pairs of $\mu = 1, 5, 10, 25, 50, 125$ and $\lambda = 5, 25, 50, 125, 250, 500$ were tested in a grid search test scenario. Obviously, combinations where μ exceeds λ were disregarded.
For variation, the ES used the CMA operator that performed best in the previous experiments without crossover. The rightmost plot in figure 4 illustrates the general trend of the μ/λ ratio. An increased λ shows a positive effect on the performance of the ES. However, the best performance



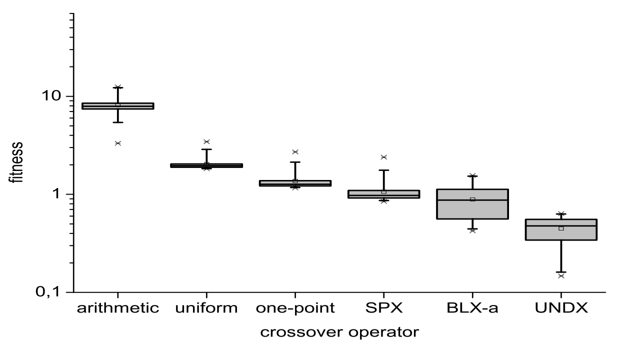
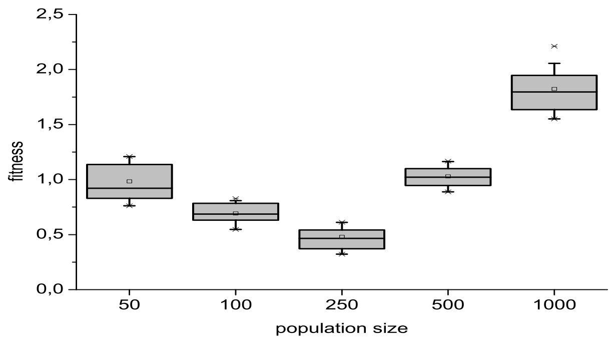


Figure 5: Impact of the crossover operators. Shown is the performance of a GA on the inference of the five-dimensional benchmark systems averaged over 20 multi-runs.



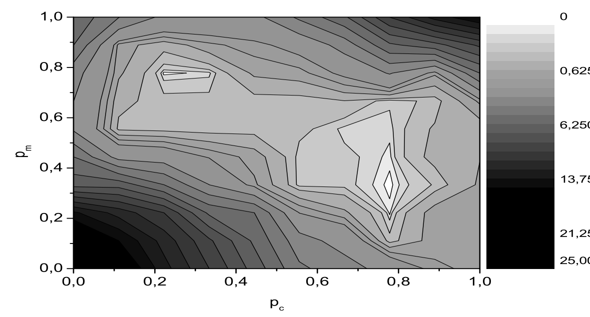


Figure 6: Impact of the population size and the optimal ratio of p_m and p_c for the best performing crossover type. Shown is the performance of a GA on the inference of the five-dimensional benchmark systems.

was achieved with a specific ratio of the two parameters: the $(\mu = 10, \lambda = 50)$-ES yielded the overall best results.

Subsumed, a $(\mu = 10, \lambda = 50)$-ES with CMA ($p_m = 1$) and no crossover performed best and was thus selected to represent the optimized ES referred to as **optES** in the following.

4.2 Optimized Genetic Algorithm

The second reference algorithm to be adjusted was a real-valued genetic algorithm, which also performed very well in the preliminary experiments. As in case of the optimized evolution strategy, the GA was evaluated with respect to the population size, crossover operators, and the optimal ratio of p_m and p_c for the best performing type of crossover.

Population Size The first experiments examined the impact of the population size on the performance of a genetic algorithm. For this purpose, five real-valued GAs using populations with 50, 100, 250, 500, and 1000 individuals were evaluated on the inference problem. The left box-plot in figure 6 shows the best-of-run fitness values averaged over 20 multi-runs. The graph shows a similar effect of the population size as in the case of the multi-start hill climber. Too small populations lead to worse fitness values in the initial generation together with an overall bad performance on the modeling problem. This results from an insufficient initial sampling and thus from a high probability of finding only unsuitable solutions in the beginning. Further on, the small population size hinders the GA that mainly relies on search space exploration, from covering the whole search space in an optimal way. On the other hand, too large population sizes reduce the performance of the GA although starting with good initial fitness values due to the smaller number of generations.

Crossover Operators To select the best performing crossover operator for a GA, several types were evaluated: arithmetic crossover [8], discrete uniform crossover [9], discrete one-point crossover [8], simplex uniform crossover (SPX) [15], BLX-α [2], and uni-modal normally distributed crossover (UNDX) [7].

UNDX and SPX used three individuals for crossover, the others only two, as suggested by the authors of the corresponding implementations, together with global mutation. Crossover was performed with a probability of $p_c = 0.8$ and mutation occurred with $p_m = 0.2$. The results given in the plots of figure 5 clearly show that the more sophisticated crossover operators – SPX, BLX-α, and UNDX – are useful for the algorithm to find good solutions. UNDX outperformed all the other operators and is thus used for further analysis to tune the reference GA.

Mutation versus Crossover Finally, the impact of the ratio of global mutation and UNDX crossover was valuated with a grid search of possible p_m/p_c combinations. As can be concluded from the right contour plot in figure 6, the GA performs worst for $p_m = p_c = 0$. Further on, the quality of the results increases with increasing mutation and crossover probabilities up to optimal combinations of

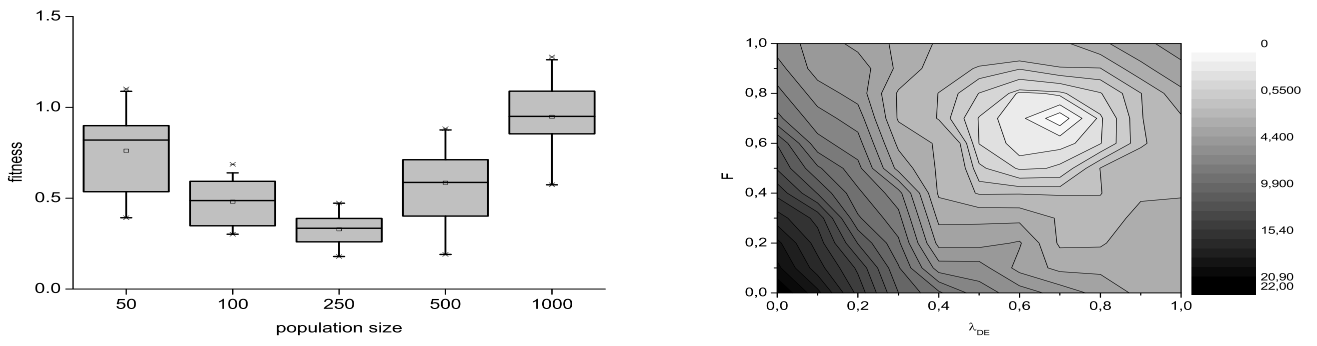


Figure 7: Impact of the population size and the optimal ratio of the adaptation weights F and λ_{DE}. Shown is the performance of an ES on the inference of the five-dimensional benchmark systems.

$p_m = 0.3$,$p_c = 0.8$ and the mirrored combination $p_m = 0.8$, $p_c = 0.3$. Interestingly, the performance of the GA drops again significantly beyond these probabilities. This is most probably due to interferences of the self-adaptation mechanisms of both operators. Overall, the GA performs best with the first mentioned ratio of $p_m = 0.3$ and $p_c = 0.8$.

Thus, the optimal setup for a GA was a real-valued GA with a population size of 250 individuals, global mutation ($p_m = 0.3$) and UNDX crossover ($p_c = 0.8$). This optimized genetic algorithm is referred to as **optGA** in the following.

4.3 Optimized Differential Evolution

The last optimization method to fine-tune was the differential evolution algorithm, which performed second best in the preliminary experiments. For the evaluation of the DE, only different population sizes and the impact of the adaptation weights F and λ_{DE} were considered, because they are known to be the most sensitive parameters of the DE.

Population Size As in case of the GA, the population was varied in size between 50 and 100 individuals ($|P| = 50, 100, 250, 500, 1000$). The box plot in figure 7 on the left shows the averaged results of 20 independent inference runs. And similar to the results of the population size analysis for the GA, the best performance was achieved for 250 individuals. And as in the case of the GA, too small populations suffered from premature convergence due to a suboptimal coverage of the search space. Too large populations, on the other hand, were not able to converge to good solutions because the resulting number of fitness evaluations is too small and thus the number of generations is insufficient.

Adaptation Weights Two parameters of the DE are very sensitive with respect to the performance of the algorithm: the adaptation weights λ_{DE} and F. The crossover probability p_c showed only minor effects on the performance in preliminary experiments and was thus not further examined. To evaluate their impact and to find an optimal setting, a grid search for both parameters was performed, varying them between 0 and 1. The results of the grid search are shown in the right plot of figure 7. Clearly, an optimal combination of λ_{DE} and F can be discerned, namely $\lambda_{DE} = F = 0.7$. This also correlates with the default parameter settings of $\lambda_{DE} = F = 0.6$ suggested by the developers of the DE.

Overall, the best setup for the optimized DE **optDE** was a combination of the adaptation weights $\lambda_{DE} = F = 0.7$, a crossover probability of $p_c = 0.9$ together with a population of 250 individuals.

Figure 8 shows the overall comparison of the best standard optimization methods (cmaES, realGA, and DE) to the optimized versions. As can be concluded from the figure, all optimized versions perform significantly better than the standard approaches. However, the individual increase in performance is varying. In case of the ES and the DE, only small performance increases can be seen, due to the minor changes of the optimized versions. However, tuning the optimized GA resulted in large increase in the performance compared to the standard real-valued GA.

5. CONCLUSIONS

In this paper, we introduced the framework JCell that was developed to allow users to evaluate different algorithms on a set of well-defined benchmark systems to obtain comparable results. Several optimization algorithms together with a variety of mathematical models are implemented to study the performance on the inference problem.

Further on, we systematically examined the performance of standard evolutionary algorithms and the impact of mutation and crossover on the problem of inferring gene regulatory networks from microarray data. The comprehensive study was performed on an Opteron cluster with 16 dualcore CPUs with 2,2GHz and 2GB RAM per node. The overall computation time amounted approximately 400h.

Additionally, we fine-tuned the three most promising algorithms, namely an evolution strategy with CMA, a real-valued genetic algorithm with UNDX, and differential evolutions, to examine their best performance. The experiments in this paper showed that although some evolutionary mechanisms like the real-valued GA with UNDX crossover and differential evolution are able to cope with the complex and highly multi-modal search space, only the CMA mutation operator solved this type of optimization efficiently and at the same time reliable. Even with additional fine-tuning of the corresponding algorithm settings for the realGA and the DE, no alternative approach was able to equal the performance of the ES with CMA with respect to the RSE. A conclusion of this paper is that to choose a suitable model and an appropriate optimization method is crucial for the inference, and the results illustrated that evolutionary algorithms are well suited for the problem of network inference.

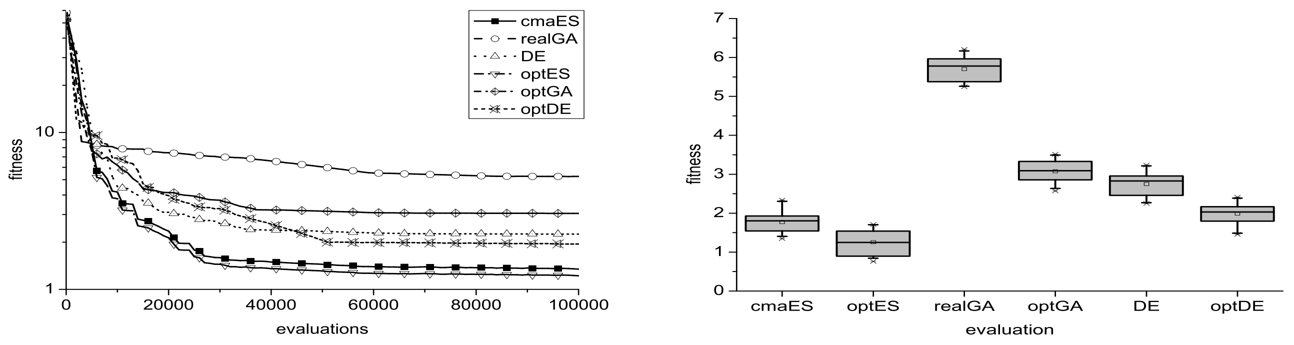


Figure 8: Comparison of the standard evolutionary methods and the optimized algorithms on the ten-dimensional benchmark systems.

In a related publication [14], we present a comprehensive study of mathematical formulations, namely weight matrices, H-systems, and S-systems for modeling dynamic systems of different mathematical properties and size.

6. ACKNOWLEDGMENTS

This work was supported by the National Genome Research Network (NGFN-II) in Germany under contract number 0313323.

7. ADDITIONAL AUTHORS

Additional authors: Jochen Supper (Centre for Bioinformatics Tübingen, Nora Speer (Centre for Bioinformatics Tübingen, and Andreas Zell (Centre for Bioinformatics Tübingen).

8. REFERENCES

[1] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, January 2002.

[2] L. Eshelman and J. Schaffer. Crossover's niche. In *Proceedings of the International Conference on Genetic Algorithms*, pages 9–14, 1993.

[3] K. Hadeler. Gedanken zur Parameteridentifikation. Personal Communication, 2003.

[4] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 312–317, 1996.

[5] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita. Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics*, 19(5):643–650, 2003.

[6] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, and A. Konagaya. Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics*, 21(7):1154–1163, 2005.

[7] H. Kita, I. Ono, and S. Kobayashi. Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 1581–1588, 1999.

[8] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer, 1992.

[9] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[10] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, 1973.

[11] M. A. Savageau. 20 years of S-systems. In *Canonical Nonlinear Modeling. S-systems Approach to Understand Complexity*, pages 1–44, 1991.

[12] H.-P. Schwefel. *Numerical optimization of computer models.* John Wiley & Sons, 1981.

[13] C. Spieth. JCell : A java framework for inferring genetic networks. Technical Report WSI-2005-07, Centre for Bioinformatics Tübingen, University of Tübingen, 2005.

[14] C. Spieth, N. Hassis, F. Streichert, J. Supper, N. Speer, K. Beyreuther, and A. Zell. Comparing mathematical models on the problem of network inference. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2006.

[15] S. Tsutsui, M. Yamamura, and T. Higuchi. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 657–664, 1999.

[16] D. Weaver, C. Workman, and G. Stormo. Modeling regulatory networks with weight matrices. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 4, pages 112–123, 1999.

[17] M. K. S. Yeung, J. Tegner, and J. J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. In *Proceedings of the National Academy of Science*, volume 99, pages 6163–6168, 2002.