Genetic Programming with Data Migration for Symbolic Regression

Michael Kommenda[1,2], Michael Affenzeller[1,2], Bogdan Burlacu[1], Gabriel Kronberger[1], Stephan M. Winkler[1]

[1]University of Applied Sciences Upper Austria
Heuristic and Evolutionary Algorithms Laboratory
Softwarepark 11, 4232 Hagenberg, Austria

[2]Johannes Kepler University Linz
Institute for Formal Models and Verification
Altenbergerstr. 69, 4040 Linz, Austria

{michael.kommenda, michael.affenzeller, bogdan.burlacu, gabriel.kronberger, stephan.winkler}@fh-hagenberg.at

ABSTRACT

In this publication genetic programming (GP) with data migration for symbolic regression is presented. The motivation for the development of the algorithm is to evolve models which generalize well on previously unseen data. GP with data migration uses multiple subpopulations to maintain the genetic diversity during the algorithm run and a sophisticated training subset selection strategy. Each subpopulation is evaluated on a different fixed training subset (FTS) and additionally a variable training subset (VTS) is exchanged between the subpopulations at specific data migration intervals. Thus, the individuals are evaluated on the unification of FTS and VTS and should have better generalization properties due to the regular changes of the VTS.

The implemented algorithm is compared to several GP variants on a number of symbolic regression benchmark problems to test the effectiveness of the multiple populations and data migration strategy. Additionally, different algorithm configurations and migration strategies are evaluated to show their impact with respect to the achieved quality.

Categories and Subject Descriptors

I.2.2 [**Automatic Programming**]: Program synthesis; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic Methods

General Terms

Algorithms, Experimentation

Keywords

Symbolic Regression, Multi-population Genetic Programming, Generalization

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2609857.

1. INTRODUCTION

Symbolic regression is the task of finding a model in the form of a mathematical expression describing the relation between a dependent variable and several independent ones as accurately as possible. Symbolic regression problems are commonly solved by genetic programming [8], a population-based and meta-heuristic optimization method for evolving a program that solves a given task. In the case of symbolic regression the program is a mathematical expression and the objective function is the correlation between the model's estimates of the dependent variable and its true values. The main advantages of symbolic regression by genetic programming are that the structure of the model does not have to be predefined, its implicit feature selection strategy [12], and the interpretability of the evolved mathematical expressions [1].

Symbolic regression is regarded as a supervised machine learning task and could produce models with weak generalization properties on unseen data. A common approach to counteract this phenomenon is to restrict the complexity of the produced models and to determine optimal algorithm parameter settings by cross-validation. However, determining optimal parameter settings is quite hard for stochastic algorithms such as symbolic regression by genetic programming, due to the occuring variations in terms of result quality for repeated algorithm executions.

Another common approach to improve the generalization properties of the produced models, which is especially suited for GP, is changing the samples used for fitness calculation regularly to avoid a too strong adaption of the models to the presented training samples. This can be done either by random subset selection [4, 5] or more specialized techniques such as coevolution [11, 6]. An approach similar to the random subset selection technique is presented in this publication and combined with a multi-population genetic programming algorithm.

Multi-population Evolutionary Algorithms

The first coarse-grained parallel and distributed evolutionary algorithms [13, 18] were developed to speed up the evolutionary process by evolving multiple populations in parallel on different machines. If no information exchange happens between the populations, the distributed algorithm is equivalent to executing multiple repetitions of a single population algorithm sequentially and aggregating the results.

One common approach to exchange information between the subpopulations is to copy or move individuals among the subpopulations according to a predefined topology at specific intervals.

The motivation behind the development of evolutionary algorithms with multiple populations, in addition to reducing the execution time, is to explore different regions of the search space simultaneously and thus, maintaining the overall genetic diversity during the algorithm execution. The genetic information present in the subpopulation is exchanged at specific intervals and this is referred to as migration. During migration a certain part of the population (e.g., the best, worst, most diverse, or random individuals) is selected and inserted in another subpopulation, where for example the worst individuals are replaced. Migration is performed according to a spatial topology of the subpopulations [2], which stays constant during the algorithm execution. Popular examples of such a topology are unidirectional or bidirectional ring topologies, but also others like grids or hyper cubes can be used.

Another advantage of multi-population evolutionary algorithms is that building blocks of the final solution can be evolved in different subpopulations and assembled by combining migrated individuals containing these building block [17, 14]. Although the concept of multiple populations for evolutionary algorithms has been originally developed for genetic algorithms, it is easily adapted to genetic programming.

The subsequent parts of the publication are organized as follows: Section 2 introduces the algorithm modifications and the resulting consequences. In Section 3 a detailed experiment plan including a description of the used benchmark problems and parameter settings is given and the obtained results are presented and discussed in Section 4. Finally, Section 5 concludes this publication and explains possible shortcomings and necessary further test and developments to improve the methodology.

2. METHODS

Multi-population evolutionary algorithms explore different regions of the search space simultaneously and individual migration maintains the population diversity during the algorithm execution [14]. Additionally, multiple populations allow an easy integration of advanced sampling strategies for training subset selection and should lead to increased generalization capabilities of the evolved models.

We have combined these ideas into a multi-population genetic programming algorithm for symbolic regression, where each subpopulation is evolved on different parts of the training data. The distribution of the training data is performed during the initialization of the algorithm and the selected training subset, denoted as *FixedTrainingSubset* (FTS), stays constant during the algorithm execution. If the reciprocal of the number of subpopulations is equal to the relative amount of the FTS, the whole training data is divided evenly (without any overlaps) across the subpopulations. Alternatively, if the size of the FTS is smaller, not every sample of the training data will be covered and if it is larger, the individual subsets will overlap. If no individual migration is included, this algorithm performs exactly the same as standard genetic programming, applied multiple times on different parts of the training data, with an aggregation of the obtained results.

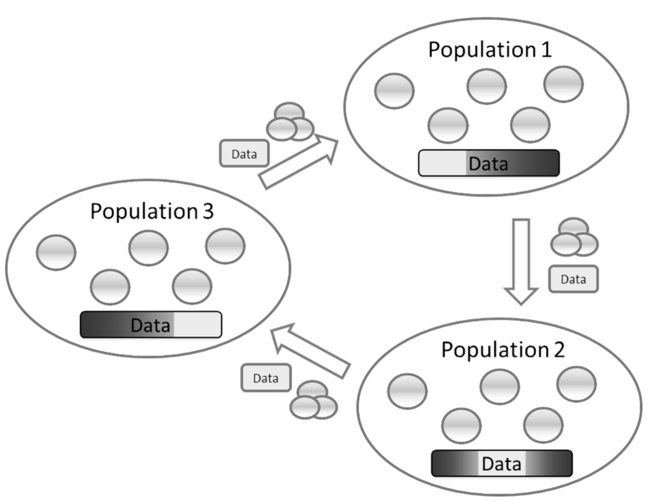

Figure 1: Multipopulation GP with data migration.

An unidirectional ring structure is used as the migration topology of the algorithm and the best individuals are moved to the clockwise neighbor of the current subpopulation.The number of immigrated and emigrated individuals are the same for each subpopulation and therefore the size of each subpopulation stays constant during the algorithm execution. The migrated individuals are evaluated on different parts of the training data and are expected to have better generalization capabilities compared to individuals evolved on a constant subset of the training partition.

A drawback of using only a fixed training subset to evaluate the individuals is that only migrated individuals are trained on different parts of the training data. Individuals that are never migrated are always evaluated on the same training subset and are more likely to generalize poorly. Another aspect is that migrated individuals have to be reevaluated on the FTS of the new subpopulation, so that selection is not mislead by outdated fitness information. The reevaluation could lead to a drastic change in the fitness and to diminish this effect a data migration step (in addition to individual migration) has been included in the multi-population algorithm.

As a result, the fitness of an individual is calculated on samples included in the FTS and an changing *VariableTrainingSubset* (VTS). The VTS is initially chosen as a consecutive part of the training data adjoining the FTS. These two subsets never overlap to avoid the repeated evaluation of training samples. At a specific data migration interval, which by default is equal to the individual migration interval, the VTS is migrated to another subpopulation. By using a fixed and variable training subset for every subpopulation the generalization properties of the evolved individuals should be increased. At the same time the genetic diversity is maintained by migrating the individuals and the algorithm should be able to produce better solutions compared to single population variants. Figure 1 depicts the algorithm, where the training data is distributed evenly across the subpopulations.

As a consequence of evaluating the fitness of an individual only on parts of the training data, the individual with the highest fitness does not have to be the best individual on the whole training data. This causes problems when the best model has to be selected as the algorithm result. Therefore, after the algorithm is finished, every individual in the final population is reevaluated on the whole training data, so that the fitness reflects the model's fit on the training partition. The algorithm steps of the multi-population genetic programming algorithm with data migration for symbolic regression are given as pseudo code in Algorithm 1.

Algorithm 1 Multi-population GP with data migration.

```
for each $Subpopulation$ do
    Assign fixed training subset $FTS$
    Assign variable training subset $VTS$
    Generate random individuals
    Evaluate individuals on $FTS \cup VTS$
end for

$generations \leftarrow 0$
while $generations < maxGenerations$ do
    for each subpopulation do
        Generate offspring from selected parents
        Evaluate offspring on $FTS \cup VTS$
        Replace parents with newly generated offspring
    end for
    if Migrate individuals? then
        Comment: Data changes for migrated individuals
        Migrate best individuals to the next subpopulation
        Reevaluate migrated individuals on new data
    end if
    if Migrate data? then
        Comment: Data changes for all individuals
        Migrate $VTS$ to the next subpopulation
        Reevaluate all individuals
    end if
    $generations \leftarrow generations + 1$
end while

Reevaluate all individuals on whole training data
return Best individual
```

3. EXPERIMENTS

We tested the multi-population GP algorithm with data migration on several synthetic benchmark problems to examine the effects regarding the generalization properties of the resulting models. The problems were taken from the recommended GP benchmark problems[1] [16] and additionally the Poly-10 [10], the Friedman-1, and the Friedman-2 [3] problems were included. A detailed description of the used benchmark problems, including the data generating formula and the number of training and test samples, is given in Table 2.

Algorithm Configurations

For the first experiment four different algorithm configurations were tested: single-population genetic programming (GP), single-population genetic programming with data migration (GP+DM), multi-population genetic programming (MultiPop GP), and multi-population genetic programming with data migration (MultiPop GP+DM). In contrast to the configurations without data migration, the ones with data migration use only a certain part of the available training data for fitness calculation.

The configurations with data migration use 20% of the available training data as fixed training subset (FTS) and 20% as variable training subset (VTS) that is altered during the algorithm execution with a data migration interval of 5 generations. Both multi-population algorithms are configured to include 5 subpopulations and an individual migration interval of 5 generations. Therefore, the MultiPop GP + DM algorithm migrates individuals and data always simultaneously.

Additionally, to the comparison of the achieved qualities of the different algorithms, the influence of data migration on the algorithm is investigated. Based on the MultiPop GP+DM algorithm, we performed tests with both, individual and data migration, with only one of the migration types, and with no migration at all between the different subpopulations. For all configurations the fixed training subset is used and contains 20% of the available training data as stated above.

[1]http://www.gpbenchmarks.org/wiki/index.php?title=Problem_Classification

Table 1: Algorithmic settings common for all algorithms.

Parameter	Value
Tree initialization	PTC2 [9]
Maximum tree length	200 Nodes
Population size	500 Individuals
Elites	1 Individual
Selection	Tournament selection
	Group size 4
Crossover probability	100%
Crossover operator	Subtree crossover
Mutation probability	25%
Mutation operators	Single point mutation
	Remove branch
	Replace branch
Fitness function	Coefficient of determination R^2
Termination criterion	100 Generations
Terminal symbols	$Constant$, $weight * variable$
Function symbols	Binary functions$(+, -, \times, /)$

Further specific algorithm settings are described textually.

Parameter settings

The parameter settings that are common to all configurations are listed in Table 1. For all configurations that use multiple subpopulations, 5 subpopulations with a size of 100 were used, so that the total number of individuals stays constant regardless of using a single or multiple populations.

The fitness function for all GP algorithms is the coefficient of determination R^2 (Equation 1), which is defined between [0,1]. The R^2 automatically takes linear transformations of the models into account to achieve the best possible fit on the data, but as a consequence the models have to be scaled linearly before other quality values (e.g., mean square error, average relative error, ...) can be calculated. The result of a GP run is the best model in the last generation evaluated on the whole training partition.

Table 2: Definition of benchmark problems and training and test ranges.

Name	Function	Training	Test
Nguyen-12	$f(x,y) = x^4 - x^3 + 0.5y^2 - y$	200 samples	1000 samples
Keijzer-14	$f(x,y) = \frac{8}{2+x^2+y^2}$	200 samples	798 samples
Vladislavleva-5	$f(x_1,...,x_3) = 30\frac{(x_1-1)(x_3-1)}{{x_2}^2(x_1-10)}$	300 samples	2700 samples
Poly-10	$f(x_1,...,x_{10}) = x_1x_2 + x_3x_4 + x_5x_6 + x_1x_7x_9 + x_3x_6x_{10}$	250 samples	250 samples
Pagie-1	$f(x,y) = \frac{1}{1+x^{-4}} + \frac{1}{1+y^{-4}}$	676 samples	1000 samples
Friedman-1	$f(x_1,...,x_{10}) = 0.1\exp(4x_1) + \frac{4}{1+\exp(-20(x_2-0.5))} + 3x_3 + 2x_4 + x_5 + \text{Noise}$	500 samples	5000 samples
Friedman-2	$f(x_1,...,x_{10}) = 10\sin(\pi x_1x_2) + 20(x_3 - 1/2)^2 + 10x_4 + 5x_5 + \text{Noise}$	500 samples	5000 samples

$$R^2(x,y) = \frac{\text{Cov}(x,y)^2}{\text{Var}(x) * \text{Var}(y)} \quad (1)$$

All the benchmark problems, algorithms and extensions in this publication are implemented in HeuristicLab [15], an open source framework for heuristic optimization.

4. RESULTS

The results presented in this section were aggregated over 50 repetitions for every algorithm configuration to take the stochasticity of the algorithms into account and report the coefficient of determination R^2 on the training and test data.

Table 3 reports the median, the average, and the standard deviation of the quality of the best models generated by repeated algorithm executions on all benchmark problems. In the first two columns standard genetic programming (GP) using the whole training data is compared to single-population genetic programming with data migration (GP+DM), where the first 20% of the training data are used as fixed training subset (FTS) and another 20% are used as variable training subset (VTS). The last two columns show the achieved qualities of multi-population genetic programming without (MultiPop GP) and with data migrations (MultiPop GP+DM), where the same parameters for data migration were applied as in the single-population case.

When comparing GP with GP+DM, the training results produced by GP are slightly better, but in turn the better test results are obtained by GP+DM. The largest differences occur on the Poly-10 and Pagie-1 datasets, where neither of the two algorithms was able to find any model describing the data generating formula very well. The obtained results with multi-population algorithms are in general better compared to the single-population ones (e.g., Nguyen-12, Pagie-1, Friedman-1).

The picture for the influence of the data migration on the multi-population algorithm variants is similar to the single-population ones. Again, the training results are better without data migration and the test results are better with data migration, but these differences are rather small. Although the improvements in terms of the quality for the data migration algorithms are not very high, it is interesting to observe that apparently it is not necessary to use all available training samples for the fitness of an individual during the algorithm execution and still well-fitting models are created.

Migration strategies

After this first experiment we investigated the influence of the different migration strategies incorporated in the algorithm. Table 4 summarizes the obtained results (median, average and standard deviation) for the multi-population algorithm with four different configurations, where 20% of the training samples were used as FTS. The first column shows the results for the MultiPop GP+DM runs, in the second column the algorithm was configured to use no data migration (VTS=0%), but individual migration and vice versa in the third column (VTS=20%, no individual migration). In the last column the results are stated with no migration at all, which is equivalent to repeating a single-population algorithm for every different FTS and reporting the best model on the whole training of all repetitions.

The worst results when using only a subset of the available data for each subpopulation were obtained by using no migration at all. The picture is not as clear when only data or individual migration are used, but in general it is slightly better to use data instead of individuals migration. However, the best performing strategy is the MultiPop GP+DM with data and individual migration, especially on the harder problems (Poly-10 - Frieman-2).

5. DISCUSSION & OUTLOOK

In this publication a new multi-population genetic programming algorithm for symbolic regression is presented. The main innovation is that every subpopulation is evolved on a different, fixed training subset (FTS) and a changing variable training subset (VTS). Additionally, the VTS is exchanged between the subpopulations by migration according to an unidirectional ring topology. The intended benefits of the developments were that multiple populations promote the genetic diversity and hence, the algorithm learns better on the presented data. At the same time data migration should prevent the algorithm from producing overfit models by changing the data used for fitness calculations and thus, models with better generalization properties should be evolved. Another benefit of using only a subset of the training data for fitness evaluation is that the execution time of the algorithm is reduced, which is important when dealing with large datasets.

The shown results obtained by the new algorithm are encouraging in the sense that the results are as good or slightly better as the results obtained by multi-population genetic programming and for the Pagie-1 benchmark problem the multi-population GP with data migration clearly outperformed all other tested algorithm variants. However, the presented algorithm should be tested on further problems to highlight its advantages and drawbacks. Furthermore, the influence of different parameter settings, especially regarding the impact of the sizes of the two training subsets, whether individual and data migration should be performed simultaneously or asynchronously, and other possible migration topologies is subject to further research.

Another interesting idea for further developments is to combine data migration with offspring selection genetic programming and constant optimization [7], because this algorithm variant achieves better results on symbolic regression problems compared to standard GP. A drawback of the enhanced learning abilities is that the algorithm tends to adapt on the presented data to well and thus, it is more likely that overfit models are produced. Hence, data migration could be an appropriate extension to diminish these effects.

6. ACKNOWLEDGMENTS

The work described in this publication was partly done within the Josef Ressel Centre for Heuristic Optimization *Heureka!* sponsored by the Austrian Research Promotion Agency (FFG).

7. REFERENCES

[1] M. Affenzeller, S. Winkler, G. Kronberger, M. Kommenda, B. Burlacu, and S. Wagner. Gaining deeper insights in symbolic regression. In R. Riolo, J. H. Moore, and M. Kotanchek, editors, *Genetic Programming Theory and Practice XI*, Genetic and Evolutionary Computation. Springer, 2014.

[2] E. Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171, 1998.

[3] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[4] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving from Nature-PPSN III*, pages 312–321. Springer, 1994.

[5] I. Goncalves, S. Silva, J. B. Melo, and J. M. B. Carreiras. Random sampling technique for overfitting control in genetic programming. In A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, editors, *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 218–229, Malaga, Spain, 11-13 Apr. 2012. Springer Verlag.

[6] R. Harper. Spatial co-evolution in age layered planes (SCALP). In *IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, 18-23 July 2010. IEEE Press.

[7] M. Kommenda, G. Kronberger, S. Winkler, M. Affenzeller, and S. Wagner. Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion (GECCO2013)*, pages 1121–1128. ACM, 2013.

[8] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[9] S. Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, 4(3):274–283, Sept. 2000.

[10] R. Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In *Proceedings of the 6th European conference on Genetic programming*, EuroGP'03, pages 204–217, Berlin, Heidelberg, 2003. Springer-Verlag.

[11] M. D. Schmidt and H. Lipson. Coevolution of fitness predictors. *Evolutionary Computation, IEEE Transactions on*, 12(6):736–749, 2008.

[12] S. Stijven, W. Minnebo, and K. Vladislavleva. Separating the wheat from the chaff: on feature selection and feature importance in regression random forests and symbolic regression. In S. Gustafson and E. Vladislavleva, editors, *3rd symbolic regression and modeling workshop for GECCO 2011*, pages 623–630, Dublin, Ireland, 12-16 July 2011. ACM.

[13] R. Tanese. Distributed genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 434–439. Morgan Kaufmann Publishers Inc., 1989.

[14] M. Tomassini, L. Vanneschi, F. Fernández, and G. Galeano. A study of diversity in multipopulation genetic programming. In *Artificial Evolution*, pages 243–255. Springer, 2004.

[15] S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, and M. Affenzeller. *Advanced Methods and Applications in Computational Intelligence*, volume 6 of *Topics in Intelligent Engineering and Informatics*, chapter Architecture and Design of the HeuristicLab Optimization Environment, pages 197–261. Springer, 2014.

[16] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaskowski, U.-M. O'Reilly, and S. Luke. Better GP benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1):3–29, Mar. 2013.

[17] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–48, 1999.

[18] D. Whitley and T. Starkweather. Genitor ii: A distributed genetic algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(3):189–214, 1990.

Table 3: Training and test qualities given as the coefficient of the determination R^2 between the estimates of the best training solution returned by the algorithm and the target values. The shown values are the median (first row), average (second row), and standard deviation (third row) of 50 repetitions of every algorithm variant.

	GP		GP + DM		MultiPop GP		MultiPop GP + DM	
	Training	**Test**	Training	**Test**	Training	**Test**	Training	**Test**
	0.993	**0.991**	0.990	**0.987**	0.992	**0.989**	0.994	**0.993**
Nguyen-12	0.993	**0.902**	0.990	**0.939**	0.992	**0.922**	0.993	**0.958**
	0.004	**0.200**	0.005	**0.129**	0.004	**0.175**	0.004	**0.146**
	0.993	**0.987**	0.992	**0.987**	0.996	**0.992**	0.995	**0.989**
Keijzer-14	0.991	**0.890**	0.989	**0.920**	0.993	**0.931**	0.992	**0.965**
	0.008	**0.218**	0.009	**0.185**	0.007	**0.189**	0.007	**0.092**
	0.982	**0.946**	0.995	**0.989**	0.994	**0.989**	0.997	**0.994**
Vladislavleva-5	0.966	**0.808**	0.962	**0.878**	0.978	**0.904**	0.982	**0.897**
	0.042	**0.291**	0.058	**0.216**	0.031	**0.209**	0.036	**0.233**
	0.503	**0.247**	0.403	**0.230**	0.521	**0.284**	0.517	**0.475**
Poly-10	0.537	**0.311**	0.500	**0.320**	0.559	**0.347**	0.531	**0.416**
	0.123	**0.207**	0.165	**0.241**	0.149	**0.255**	0.167	**0.251**
	0.866	**0.271**	0.651	**0.207**	0.902	**0.386**	0.507	**0.456**
Pagie-1	0.839	**0.456**	0.680	**0.346**	0.862	**0.465**	0.596	**0.473**
	0.095	**0.370**	0.176	**0.333**	0.085	**0.338**	0.263	**0.339**
	0.754	**0.760**	0.735	**0.739**	0.768	**0.780**	0.754	**0.776**
Friedman-1	0.755	**0.685**	0.730	**0.666**	0.761	**0.734**	0.751	**0.759**
	0.016	**0.141**	0.034	**0.158**	0.768	**0.105**	0.018	**0.060**
	0.799	**0.759**	0.785	**0.744**	0.809	**0.781**	0.795	**0.764**
Friedman-2	0.801	**0.735**	0.787	**0.717**	0.812	**0.761**	0.795	**0.750**
	0.042	**0.102**	0.039	**0.113**	0.033	**0.074**	0.029	**0.071**

Table 4: Training and test qualities given as the coefficient of the determination R^2 between the estimates of the best training solution returned by the algorithm and the target values. The shown values are the median (first row), average (second row), and standard deviation (third row) of 50 repetitions of the algorithm with different migration strategies.

	MultiPop GP + DM		No data migration		No individual migration		No migration	
	Training	**Test**	Training	**Test**	Training	**Test**	Training	**Test**
	0.994	**0.993**	0.990	**0.984**	0.991	**0.991**	0.986	**0.980**
Nguyen-12	0.993	**0.958**	0.987	**0.943**	0.991	**0.973**	0.983	**0.914**
	0.004	**0.146**	0.008	**0.148**	0.004	**0.069**	0.010	**0.187**
	0.995	**0.989**	0.991	**0.980**	0.992	**0.989**	0.982	**0.974**
Keijzer-14	0.992	**0.965**	0.960	**0.865**	0.991	**0.982**	0.957	**0.835**
	0.007	**0.092**	0.123	**0.258**	0.005	**0.029**	0.093	**0.278**
	0.997	**0.994**	0.986	**0.960**	0.984	**0.942**	0.962	**0.871**
Vladislavleva-5	0.982	**0.897**	0.954	**0.851**	0.971	**0.807**	0.941	**0.745**
	0.036	**0.233**	0.056	**0.218**	0.031	**0.295**	0.062	**0.299**
	0.517	**0.475**	0.303	**0.199**	0.472	**0.262**	0.221	**0.055**
Poly-10	0.531	**0.416**	0.341	**0.243**	0.483	**0.323**	0.217	**0.110**
	0.167	**0.251**	0.156	**0.206**	0.134	**0.225**	0.157	**0.162**
	0.507	**0.456**	0.496	**0.429**	0.849	**0.607**	0.454	**0.165**
Pagie-1	0.596	**0.473**	0.474	**0.338**	0.739	**0.562**	0.323	**0.221**
	0.263	**0.339**	0.125	**0.192**	0.184	**0.333**	0.201	**0.206**
	0.754	**0.776**	0.715	**0.736**	0.741	**0.760**	0.708	**0.712**
Friedman-1	0.751	**0.759**	0.718	**0.720**	0.740	**0.752**	0.699	**0.614**
	0.018	**0.060**	0.031	**0.085**	0.028	**0.043**	0.053	**0.191**
	0.795	**0.764**	0.753	**0.733**	0.775	**0.742**	0.737	**0.707**
Friedman-2	0.795	**0.750**	0.761	**0.723**	0.780	**0.721**	0.736	**0.616**
	0.029	**0.071**	0.040	**0.089**	0.031	**0.093**	0.051	**0.179**