Treating Noisy Data Sets with Relaxed Genetic Programming

Luis Da Costa[1,2], Jacques-André Landry[1], and Yan Levasseur[1]

[1] LIVIA, École de Technologie Supérieure, Montréal, Canada
[2] INRIA Futurs, LRI, Univ. Paris-Sud, Paris, France

Abstract. In earlier papers we presented a technique ("**RelaxGP**") for improving the performance of the solutions generated by **G**enetic **P**rogramming (GP) applied to regression and approximation of symbolic functions. RelaxGP changes the definition of a *perfect* solution: in standard symbolic regression, a perfect solution provides exact values for each point in the training set. RelaxGP allows a perfect solution to belong to a certain interval around the desired values.

We applied RelaxGP to regression problems where the input data is noisy. This is indeed the case in several "real-world" problems, where the noise comes, for example, from the imperfection of sensors. We compare the performance of solutions generated by GP and by RelaxGP in the regression of 5 noisy sets. We show that RelaxGP with relaxation values of 10% to 100% of the gaussian noise found in the data can outperform standard GP, both in terms of generalization error reached and in resources required to reach a given test error.

1 Motivation and Background

Darwinian evolution and Mendelian genetics are the inspiration for the field of **G**enetic **P**rogramming (GP); GP solves complex problems by evolving populations of computer programs, and is particularly relevant in approaching non-analytical, multiobjectives and (practically) infinite solution space dimension problems (for details on GP, we invite interested readers to consult [6] or [1]). GP has proved its utility for supporting human analysis of problems in a variety of domains: computational molecular biology [7], cellular automata, sorting networks, analogical electrical circuits [5], among them.

We showed, in [2] and [3], that calculating fitness values with "relaxed" points on a GP problem helps avoiding the overfitting of the best solutions, as well as reducing the use of resources on the course of a GP run. Specifically, we designed a variation of GP where a *perfect* solution is not defined as the solution whose training points are reached with error zero, but as a solution that is *close enough* to them. In practical terms, this means that the fitness calculation is modified: a solution has fitness 0 ("perfect") if its approximation falls within an interval around the real targeted values. This concept is depicted in Fig. 1; the original ("perfect") point is (x_1, y_1). Now, with an absolute relaxation value of δ, any point with y value in $(y_1 - \delta, y_1 + \delta)$ has a perfect (zero) fitness.

N. Monmarché et al. (Eds.): EA 2007, LNCS 4926, pp. 1–12, 2008.

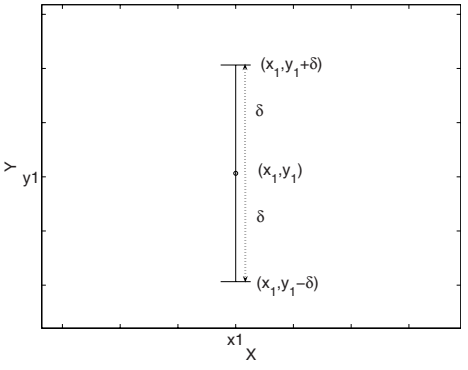
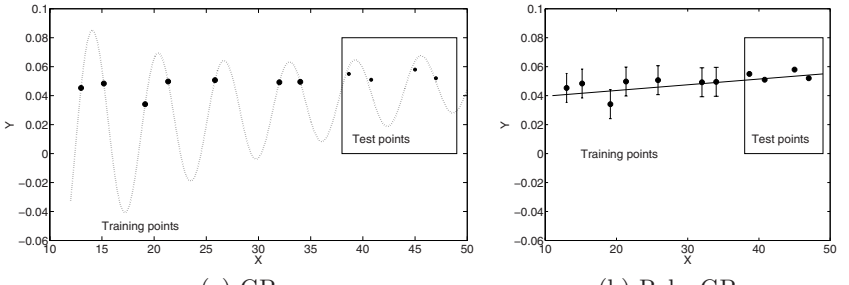


Fig. 1. Relaxation (δ) at one point. For fitness matters, the distance between (x_1, y_1) and (x_1, y_2) is 0 if $y_2 \in [y_1 - \delta, y_1 + \delta]$, $|y_2 - y_1|$ otherwise.

This definition enables a "relaxation" of the data, as the selection of individuals over the genetic process is more permissive. As an example, in Fig. 2, GP was used to evolve an individual well-trained to a certain training set (Fig. 2(a)); that best individual resulted in a sinusoidal function. On the other hand, a population evolved by RelaxGP yielded a straight line as best individual (the relaxation is noticeable by the vertical segments at each training point); a straight line being a simpler mathematical model than a sinusoidal function, the best individual generated by RelaxGP is then simpler than the one generated by GP. Their test errors are similar (right part of Figures 2(a) and (b)).



(a) GP (b) RelaxGP

Fig. 2. Solutions by standard and relaxed GP for a problem. The sinusoidal function (left) was produced by standard GP, and corresponds to $sin(x)/x+x/1000$. The straight line was produced by relaxed GP.

Our basic working hypothesis is that relaxing the data set gives more freedom to the solutions generated by GP, avoiding in this way overfitting to the training set, and guiding the population towards a generalizing solution. At the same time, giving some latitude to the individuals in a population produces more compact solutions than with regular fitness calculation.

In this paper we start from the idea that the proposed RelaxGP technique should be very useful in treating data that is, from the beginning, noisy and uncertain. We feel it is counterproductive to try to reach a perfect fit on points that are not perfect, by themselves. This is the case for most practical applications, where uncertainty in measurements is always present, and is considered part of the problem by the researchers.

To measure the adequacy of the technique for solving this kind of problems, we compared the performance of regular GP with those of RelaxGP (with various relaxations) on the task of predicting the output of a noisy set of points. These points were generated by introducing Gaussian noise to the output of the "quartic polynomial function" ($Q(x) = x^4 + x^3 + x^2 + x$).

In the remaining of the paper, we present our experimental protocol and experiments; our main objective is to find a relation between the quantity of gaussian noise in data and the best relaxation value for RelaxGP. This allows for an appropriate relaxation value to be chosen if gaussian noise can be found and measured in a problem's input data. We also show how the RelaxGP technique can outperform standard GP in terms of both the (computational) resources required for the experiments and the actual test performance when the optimal relaxation value is used.

2 Experimental Setting

2.1 Sets of Noisy Points

Gaussian Noise. A point y, perturbed by an amount of Gaussian noise a, yields a value y'; y' belongs to the interval $[y-a, y+a]$ with probability .95. a is called an *absolute* noise (a can be seen as approximately twice the standard deviation of the data).

A *relative* noise r is defined with respect to the total range δY of values $Q(x)$ can take when $x \in [-10, 10]$. The absolute value corresponding to r is $a = (r * \delta Y)/100$. For $x \in [-10, 10], \delta Y = 1.111 * 10^4$

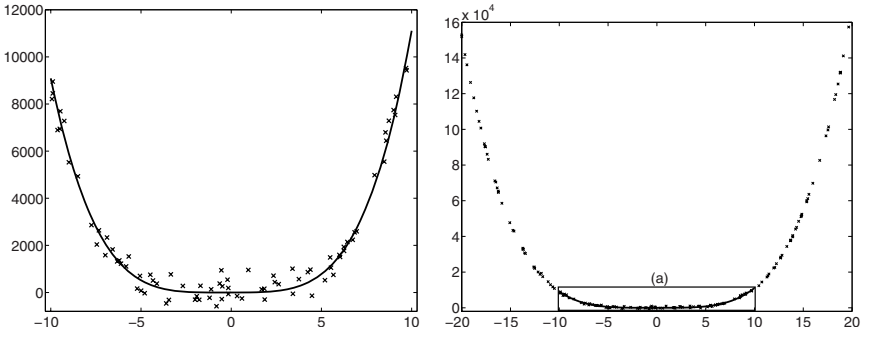
For this paper, we used 5 relative noise values: 0.5, 1, 2, 5 and 10.

Training Points. We generated several sets of "noisy" points used for training. Each set has 80 points in the region $x \in [-10, 10]$ (one set for each noise value), generated from the mathematical definition of the quartic polynomial ($Q(x) = x^4+x^3+x^2+x$) and introducing the given amount of Gaussian noise. In Fig. 3(a) the 80 points generated for training with noise 10% are shown.

Test Points. 160 noisy points were generated for testing, for each noise value, in the region on interval $x \in [-20, 20]$. In Fig. 3(b) the points generated for testing with noise 10% are shown.

2.2 RelaxGP: Relaxation Values

We applied an amount of relaxation relative to the noise present in the points of the set. In that way, different experiments, with different noises, can be compared



(a) Training (b) Test

Fig. 3. Values from the quartic function. 80 values were generated for training in the interval $x \in [-10, 10]$ for each noise value. 160 values were generated for testing in the interval $x \in [-20, 20]$ for each noise value. The values generated with noise 10% are shown as crosses. The full line is the plot of $Q(x)$. Sub-plot (a) was zoomed from sub-plot (b).

to each other. For a given (relative) noise the amounts of relaxation applied to the fitness functions were: 0 (regular GP), 10%, 100% (so relaxation = noise), 200% and 500%.

2.3 GP (and RelaxGP) Runs' Parameters

500 runs for each experiment were conducted, with the following parameters for each run (for details on the meaning of each parameter, please refer to [6]):

- Population of 500 individuals.
- Roulette selection reproduction.
- Crossover probability: 95%
- Mutation rate = 10%.
- Elitism defined by selection after reproduction: children are generated from the population of parents, and from this new population (parents and children together) only the best 500 are kept.
- A run stops when either (a) the best solution of the population solves the success predicate or (b) generation 50 is reached.

The experiments were conducted with **GPLAB** [8], a GP toolbox for **MATLAB**, modified in order to allow optimization by intervals.

2.4 Measures

In this problem we're studying resource utilization and generalization power.

Resource utilization. We presented in [3] a complete method for measuring the performance of the genetic programming paradigm in terms of the amount of

computer processing necessary to solve a particular problem. This is an extension of the method presented by Koza in [6, chap. 8]. A quick refresher is presented here.

The need for a careful inspection of the results of GP runs arises from the fact that there is randomness in the normal functioning of the GP algorithm: in the creation of the initial population, selection of individuals for reproduction, selection of crossover points, number of genetic operators to be executed and (possibly) the election of the points where fitness is calculated. Because of these probabilistic steps, there is no guarantee that a given run will yield an individual that satisfies the success predicate of the problem after being run for a number of generations [6, page 191].

One way to minimize this effect is to do multiple independent runs; the amount of computational resources required by GP[1] is then determined by (a) the number of independent runs needed to yield a success with a certain probability p and (b) the resources used for each run. The total number of resources is calculated as the sum of all resources used on each of the runs.

The method presented in [3] and [6, chap. 8] aims at estimating, in a robust statistical way, the amount of processing needed to reach a certain success predicate with a certain probability. For that we perform an important number of replications (500, in [3]) and we calculate (1) the practical probability of reaching the predicate at a certain generation, and (2) the amount of resources needed to reach a certain generation. Combining these two pieces of information yields the best couple (k, g), indicating the need to replicate the experiment k times up to generation g to reach the objective.

The result of such an analysis produces a curve and a table of values (see Fig. 4 and the accompanying Table). We can observe on Fig. 4 that the number of nodes to be evaluated is minimized when running 5 replications up to generation 40. The objective would then be reached at the evaluation of $1.63 * 10^6$ nodes.

Generalization Power. A central problem in the field of *statistical pattern recognition* is how to design a classifier that could, based on a set of examples (the *training set*), suggest actions when presented with *novel* patterns[4]. This is the issue of *generalization*.

Since good generalization is our objective, our data will be split in two parts: one is used as the traditional training set for designing the approximating function. The other (the "test" set) is used to estimate the generalization error. As mentioned in Subsec. 2.1, the training set for each experiment has 80 points on the interval $[-10, 10]$, the test set has 160 points on the interval $[-20, 20]$.

3 Results

For each noise we performed 6 groups of experiments, each with a relaxation relative to the noise. Each group corresponds to a relaxation of 10%, 50%, 100%,

[1] Or by the conventional genetic algorithm.

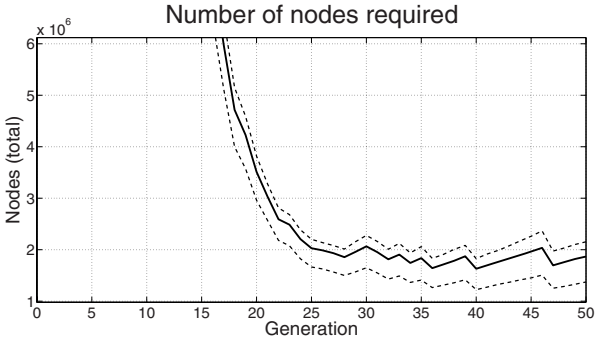


Fig. 4. Total resources (nodes) to evaluate to reach a given objective. The solid line is the average of the value, the dotted lines correspond to average ± standard deviation. The values are presented in the following table:

Generation	Runs	Number of required nodes (10^6)	Generation	Runs	Number of required nodes (10^6)
20	31	3.51	40	5	1.63
36	6	1.64	46	5	1.70
39	6	1.87	47	4	1.76

200% and 500%. For example, for noise 0.5, the points were relaxed a relative amount of 0.05, 0.25, 0.5, 1, and 2.5

For each experiment we measured, at each generation g:

1. **The test error for the best individual at** g: the test error for an approximation f to Q is calculated on the (160) test points as:

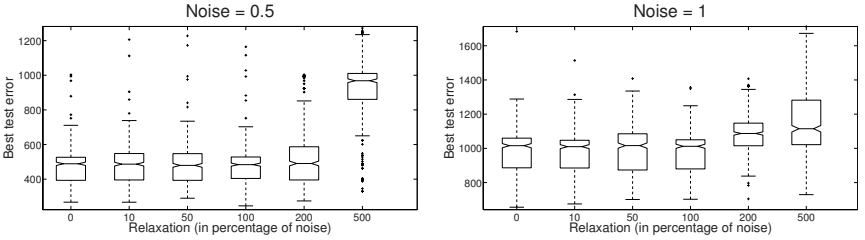
$$t_a = \sum_{i=1}^{160} |f(x_t^{(i)}) - y_t^{(i)}| \tag{1}$$

   where $(x_t^{(i)}, y_t^{(i)})$ is the i-th test point.
2. **The number of nodes evaluated to reach** g

The first results are presented as boxplots showing the values obtained for the test error of the best individual after 50 generations. Results for noise 0.5 are presented in Fig. 5(a), for noise 1 in Fig. 5(b), for noise 2 in Fig. 6(a), for noise 5 in Fig. 6(b) and for noise 10 in Fig. 7.

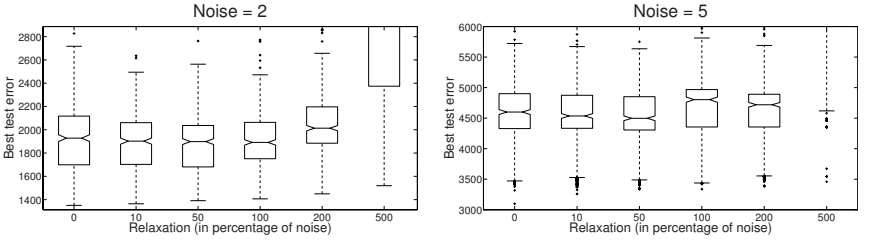
We observe that, for all noise values, the best results are obtained when the value of the relaxation is less than the noise itself (relaxation $\leq$ 100 on all boxplots). It is interesting to notice that standard GP (relaxation 0 on boxplots) does not (statistically) outperform RelaxGP with relaxation $\leq$ 100.

To obtain more precise information we show the results of the study concerning resource utilization coupled with performance values (see 2.4, page 4 for details).



(a) Noise 0.5% (b) Noise 1%

Fig. 5. Noise 0.5% and 1%



(a) Noise 2% (b) Noise 5%

Fig. 6. Noise 2% and 5%

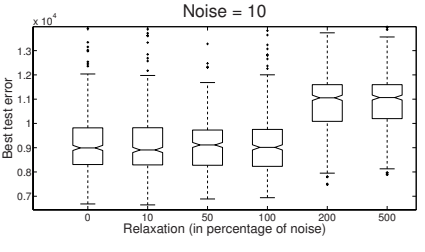


Fig. 7. Noise 10%

Comparing target errors with resources. We are interested in measuring the number of resources required to reach a certain test error t, for a certain relaxation r. For this we define a success criteria $\mathcal{S}$:

$$\mathcal{S} : \text{“}\textit{to reach test error } t\text{”}$$

We choose to look for the values that reach $\mathcal{S}$ with probability .99. The analysis of Subsec. 2.4 is then undertaken, and then the minimum value of nodes, n, reaching $\mathcal{S}$, is kept. The pair (t, n) is then interpreted as:

> *With probability* .99 *(based on 500 runs), RelaxGP with relaxation* r *needs* n *nodes to reach error* t.

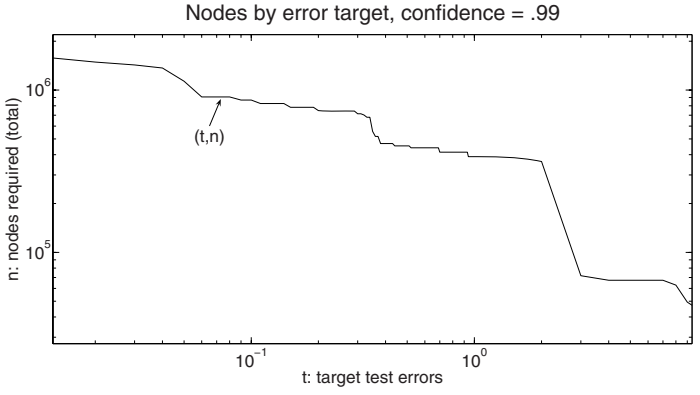


Fig. 8. Typical extended curve for a given relaxation r

Repeating this procedure for a set of test errors t yields a curve for relaxation r (Fig. 8). A point (t, n) is read as "*replications of experiment with relaxation r having reached a test error of* **at most** t *have evaluated n nodes*"; two points resulting from such calculations have been linked by a straight line.

For comparing the effects of two relaxation values r_1 and r_2 we plot the two curves together (Fig. 9). Certain comparisons can then be made; for example

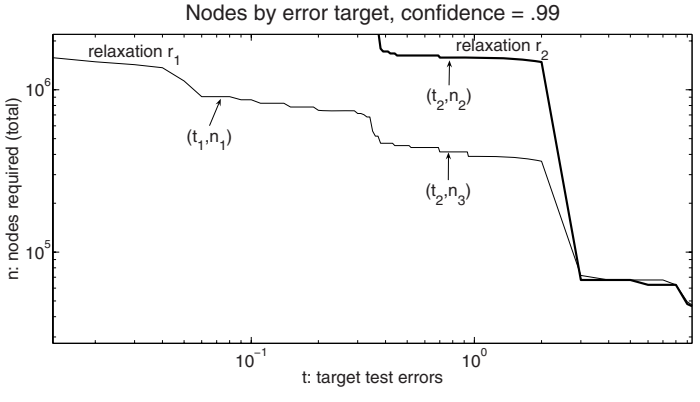


Fig. 9. Extended curve for relaxations r_1 and r_2

the curve for relaxation r_1 shows a set of points whose value can not be reached by the curve of relaxation r_2. Test error t_1 can not be reached with relaxation r_2, while with relaxation r_1 error t_1 is reached using, in average, n_1 nodes.

The relative position of the curves is also important; for example, we observe that for test error t_2 relaxation r_1 uses n_3 nodes, while relaxation r_2 uses n_2 nodes, and $n_2 > n_3$. So, if we are interested in using *less resources* for reaching error t_2, we should relax the training data by r_1 (instead of by r_2).

This idea is now being applied to the comparison of all the relaxations together. In order to compare different experiments, with different values, an absolute error t_a (defined in (1), page 6) is expressed as a percentage of (a) the number of test points and (b) the total range δY of values $Q(x)$ (the quartic polynomial) when $x \in [-20, 20]$ (interval from where the test points were generated). So, t_r, the relative test error corresponding to t_a, is:

$$t_r = \frac{t_a}{160 * \delta Y} * 100 \tag{2}$$

The analysis of the results yields several interesting points. First, for all sets of noisy values there are relaxations that outperform standard GP (*i.e.*, non-zero relaxations resulted in a best relative test error than relaxation 0). The best values for each noise are depicted in Table 1 (see Fig. 10 and Fig. 11 for details). An interesting fact from this Table is that relaxations 10% and 100% are always the best performers.

We also noticed that non-zero relaxations are generally better (in terms of resources required to reach a certain error) than standard GP in all experiments (Table 2), the only exception being a range around relative error 3.25 in noise 0.5% (Fig. 10(a)). In other words, relaxing the fitness function results in a better utilization of resources for reaching a given test error. The relative gain is the smallest in the case of the smaller noise (0.5, a gain of 12.57%), and comparable in the other cases (in the range $[22 - 28]\%$).

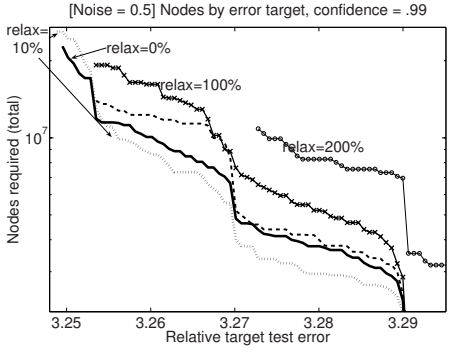
These and other results of our analysis are now presented in terms of resources required to reach a set of target **relative** errors: noises 0.5 and 1 are presented

Table 1. Best relative error reached

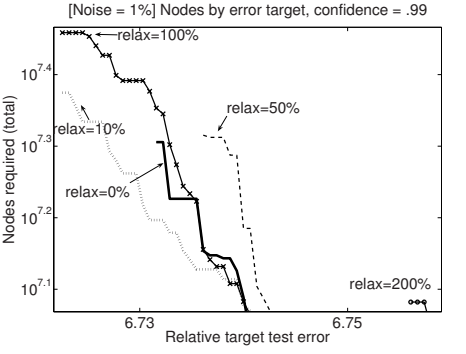
Noise	Standard GP	RelaxGP	
	Test Error	Best relaxation	Test Error
0.5	3.258	10%	3.257
1	6.731	10%, 100%	6.724
2	11.14	100%	11.11
5	33.70	10%	33.62
10	52.12	100%	51.76

Table 2. Utilization of resources

Noise	Relative error	RelaxGP		Standard GP.	Gain
	for measure	Relaxation	Resources	Resources	(%)
0.5	3.26	10%	$8.83 * 10^6$	$1.01 * 10^7$	12.57%
1	6.73	10%	$1.57 * 10^7$	$2.02 * 10^7$	22.17%
2	11.14	100%	$1.20 * 10^7$	$1.63 * 10^7$	26.38%
5	33.70	10%	$2.06 * 10^7$	$2.86 * 10^7$	27.97%
10	52.12	50%	$8.13 * 10^6$	$1.05 * 10^7$	22.57%

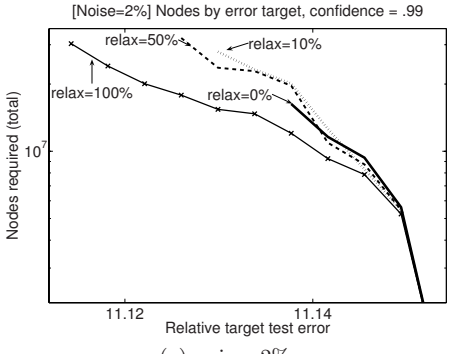


(a) noise=0.5%

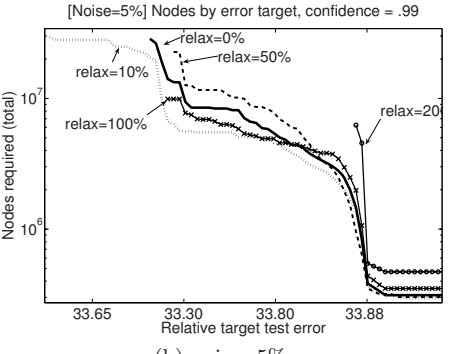


(b) noise=1%

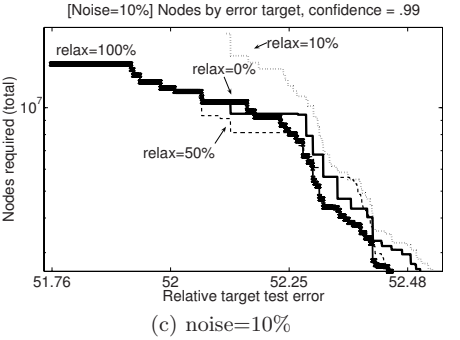
Fig. 10. Noises 0.5% and 1%



(a) noise=2%



(b) noise=5%



(c) noise=10%

Fig. 11. Noises 2%, 5%, and 10%

in Fig. 10. Noise 2, 5 and 10 are in Fig. 11. In all plots, the thick black line corresponds to the standard GP. For being considered into the plot, a target error of value t must have been reached by at least 30 of the (500) replications.

An observation made from the plots reaffirms the results presented earlier in the boxplots (Figs. 5, 6 and 7): only relaxations equal or less than the noise are competitive with standard GP: curves of relaxations 200% and 500% are worse than relaxation 0 both in best error reached and in terms of resources used.

4 Discussion

In this paper we proposed the use of a technique we developed earlier (RelaxGP, in [2] and [3]) as an alternative to treat noisy data sets in the context of regression and approximation of symbolic functions. RelaxGP stands on a new definition of a *perfect* solution: in standard symbolic regression, a perfect solution provides exact values for each point in the training set. RelaxGP allows a perfect solution to belong to a certain interval around the desired values.

Our main hypothesis was that RelaxGP should outperform classical GP in the solving of regression problems where the input data is originally noisy. Noisy data is actually found in several "real-world" problems, where the noise comes, for example, from the imperfection of sensors. We compare the performance of solutions generated by GP and by RelaxGP in the regression of 5 noisy sets. The performance was assessed through the measure of the solutions' *generalization error* (cumulative error on a set of values not used for training) and the amount of resources (in number of nodes) used for attaining a certain performance.

For all our experiments, RelaxGP, using an appropriate relaxation value, outperformed standard GP, both in terms of generalization error reached and in number of resources required to reach a certain given test error. Moreover, the amount of relaxation "optimal" for each noise n is experimentally discovered to be between 10 and 100% of the noise of the data (always assuming a Gaussian noise). In other words, if we can have an idea of "how large" the noise of the measures making up the input to a problem is, the best way to attack that problem is by using RelaxGP with relaxation lower than this noise.

These results are positive and motivate the need for another set of experiments, where we could understand more precisely the exact nature of the influence of relaxation on the dynamics of the evolutionary process, in general, and on the "cleaning" of noisy data, in particular.

Acknowledgements

The authors would like to acknowledge the financial support of the the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming An Introduction. Morgan Kauffman Publishers, San Francisco (1998)
2. Da Costa, L.E., Landry, J.-A.: Relaxed genetic programming. In: Keijzer, M. (ed.) Proceedings of GECCO 2006, Seattle, USA, pp. 937–938 (2006)

3. Da Costa, L.E., Landry, J.-A., Levasseur, Y.: Improving genetic programming by relaxing the fitness function. Improving genetic programming by relaxing the fitness function (2007) (Submitted to Genetic Programming and Evolvable Machines)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley and sons, Inc., New York (2001)
5. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.A.: Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 132–140. The MIT Press, Stanford University. Cambridge, MA (1996)
6. Koza, J.R.: Genetic Programming-On the programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge, London (1992)
7. Koza, J.R.: Evolution of a computer program for classifying protein segments as transmembrane domains using genetic programming. In: Altman, R., Brutlag, D., Karp, P., Lathrop, R., Searls, D. (eds.) Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, pp. 244–252. AAAI Press, Menlo Park, CA (1994)
8. Silva, S.: Gplab-a genetic programming toolbox for matlab (2004), `http://gplab.sourceforge.net`