SYMBOLIC REGRESSION USING TABU SEARCH IN A NEIGHBORHOOD OF SEMANTICALLY SIMILAR SOLUTIONS

Gabriel Kronberger[(a)], Andreas Beham[(b)]

[(a,b)] University of Applied Sciences Upper Austria, School of Informatics, Communications and Media

[(a)]gabriel.kronberger@fh-hagenberg.at, [(b)]andreas.beham@fh-hagenberg.at

ABSTRACT

In this paper we describe for the first time a tabu search algorithm for symbolic regression. The novel contribution presented in this work is the idea to use a metric for semantic similarity to generate moves in such a way that branches are only replaced with semantically similar branches. In symbolic regression separate parts of the solution are linked strongly; often a small random change of one part might disrupt a link and thus, can completely change the semantics of the solution. We hypothesize that by introducing the semantic similarity constraint, the fitness landscape for tabu search becomes smoother as each move can only change the fitness of the solution slightly. However, empirical evaluation on a set of simple benchmark instances shows that the approach described in this paper does not perform as well as genetic programming with offspring selection and there is no big difference between random and semantic move generation.

Keywords: symbolic regression, tabu search, genetic programming, semantic similarity

1. INTRODUCTION

Symbolic regression is a data-modeling approach for regression analysis where the structure of the model is not specified explicitly but must be discovered automatically. Thus, it can also be used in situations where no or only limited information about the modeled system is available and it would be hard to specify a fitting model structure manually. Symbolic regression has first been described in the context of genetic programming (Koza 1992) and has since become a popular benchmark task for genetic programming. However, symbolic regression has also been used successfully to solve real-world problems occurring for instance in industry, medicine and finance.

Tabu search is a trajectory-based meta-heuristic for solving difficult, usually combinatorial, optimization problems (Glover 1999). To avoid getting stuck in local optima, it uses the concept of short-term and long-term memory to achieve a good balance between diversification and intensification stages of the search process.

In this contribution we present a new approach for symbolic regression which is based on tabu search instead of an evolutionary algorithm. We describe the motivation for the algorithm and also discuss details of the implementation and necessary parameters. Additionally, we also describe the difficulties we had when transforming the initial concept into a working implementation and highlight the problems of finding good parameter values for the algorithm. An empirical evaluation of the proposed approach on a set of symbolic regression benchmark instances is also given to facilitate comparison with a genetic programming approach. The main contribution of this article is the description of a neighborhood function which produces semantically similar neighbors to improve the smoothness of the tabu search trajectory and a new semantic aspiration criterion for tabu search.

2. MOTIVATION

Symbolic regression based on genetic programming is an idea that is already more than 20 years old. However, there are still several open topics that are actively researched. These open topics are probably one of the reasons that symbolic regression still has not matured to a technology that can be used easily and routinely in practical applications. The list of open topics includes performance, overfitting, extrapolation capabilities, model complexity, and a workable theory for evolutionary operators, to name some examples. The recently proposed fast function extraction (FFX) (McConaghy 2011) side steps some of the above mentioned issues and demonstrates that symbolic regression does not necessarily have to be solved with an evolutionary algorithm. Instead it uses an approach based on regularization and coordinate descent (Hastie 2009) to search for parsimonious models by combining only relevant features from a large set of features including the original variables and (non-linear) transformations of the original variables. The advantage of this approach is its efficiency and, thus, can also be applied to large datasets and in a semi-automatic matter. Additionally, it is not necessary to tune a large number of parameters. McConaghy argues that because of these advantages it is much closer to a usable technology than genetic programming. A drawback is that the space of model structures considered by FFX is smaller than the space typically considered by GP, which also includes deeply nested functions.

FFX is a new algorithmic approach to symbolic regression that is not rooted in the evolutionary computation framework and, thus, can be advantageous in comparison to traditional GP approaches. It certainly allows a new perspective and can lead to better insights into the characteristics of the problem domain. This last aspect in particular has motivated us to look at yet another different approach based on tabu search which manipulates an initial solution using iterative moves to improve the solution step by step.

An important issue for trajectory-based algorithms is the design of a move operator for generating potential alterations of the solution that allow to efficiently walk towards a local optimum in the fitness landscape. For this it is necessary to understand the potential effects of solution alterations regarding the semantics of the whole solution. The mapping of syntactic and semantic similarity of GP solutions is generally rather chaotic and not well understood. One major issue is that disconnected parts of the solution can be strongly linked, and thus, the alteration of one branch can have a big effect on the global semantics of the whole solution even for small local alterations. Getting a better understanding of the semantic consequences of evolutionary operators is still an open research topic and recently a number of operators which try to include semantic information into the evolutionary process have been described (Nguyen 2010, Nguyen 2011, Krawiec 2011, Krawiec 2012). This has lead us to consider a semantically constrained move operator which should lead to a smoother fitness landscape.

In the following we first describe the general algorithm conceptually and then discuss the necessary details for the implementation of the algorithm. The aim is to find an efficient algorithm that is able to find accurate solutions also for hard problem instances. The main hypothesis is that tabu search using semantically similar moves improves the search, so that solutions for hard problems can be found. We do not yet discuss the generalization capabilities and parsimony of solutions in this paper.

3. PREVIOUS WORK

Variants of tabu programming that are similar to the approach discussed in this contribution have been described previously (Abdel-Rahman 2011, Balicki 2007, Balicki 2009). However, these variants use a different way to generate moves and in particular do not use a neighborhood function that leads to semantically similar neighbors and a smoother search trajectory. This extension which can be implemented rather easily for symbolic regression is described in this contribution for the first time.

4. CONCEPTUAL DESCRIPTION

Solutions are represented as symbolic expression trees in the same way as in tree-based GP. The internal nodes of the tree contain symbols of operators and functions and the leaf nodes contain input variables. The set of symbols for functions and variables is a parameter of the algorithm. Figure 1 shows an exemplary symbolic regression model.

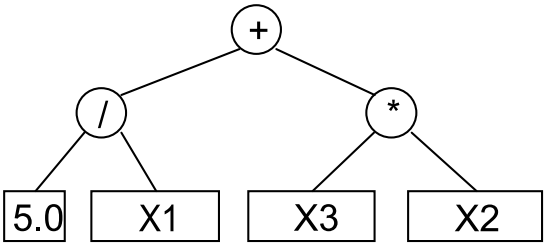

Figure 1: Example of a Symbolic Regression Model.

The proposed approach is based on the standard formulation of tabu search (Glover 1999). We use only a single tabu list for short-term memory and allow aspiration of moves. Initially a random solution is generated as a starting point. This solution is iteratively manipulated by applying the best move from a set of possible moves after checking a tabu criterion. In one move a branch of the tree is removed and replaced by a valid branch from a pool of branches. The pool of branches for replacement is filled with small random branches in the initialization step of the algorithm. We chose to use a pool of pre-generated branches to prevent continuously generating and evaluating random branches in the main loop of the algorithm. The main loop involves: generating moves, checking the tabu list, applying the best move, and updating the tabu list.

Two different modes are possible for move generation, random mode and semantic mode. In random mode, a number of cut points in the current solution are selected randomly and for each cut point a random replacement branch is selected randomly from the pool. In semantic mode, the cut points in the current solution are also selected randomly, however, then for each cut-point the semantically most similar branches from the replacement pool are selected. The semantic similarity of a branch in the current solution and a branch from the pool is determined by the squared Pearson's correlation coefficient of the output of the branch and the replacement branch. Branches with a large R^2 have a similar output as the original branch that is to be replaced and, thus, should only have a small effect on the overall quality of the model. By iteratively replacing branches with semantically similar branches the algorithm can walk on a smooth trajectory to more accurate solutions. To facilitate exploration of worse solutions the algorithm uses the tabu-list to guide search to different regions of the hypotheses space.

When a move is applied the exact point where the root of the new branch has been inserted is added to the tabu list and is thus locked and must not be changed for some time. Several aspiration criteria are used to allow certain moves that would otherwise be tabu for instance if a new best-of-run solution would be found.

A maximal size and depth limit is used to prevent uncontrolled growth of the solution.

5. DETAILED DESCRIPTION

5.1. Parameters

Based on the conceptual description of the algorithm a number of parameters can be derived. The most important parameters are the number k of randomly selected cut-points M and the number n of replacement branches for each cut-point. The total number of moves $|N|$ generated in each iteration results as the product of these two parameters. A parameter for the maximal number of iterations is necessary to stop the algorithm. The size of the tabu list determines how long a manipulated point in the tree is protected against manipulations. The size of the solution is limited by two parameters for the maximal solution length and solution depth. Additionally, the two additional parameters limit the length and depth of the replacement branches in the pool and another parameter is necessary for the size p of the pool F.

5.2. Algorithm Description

1. Create an initial solution (tree) randomly.
2. Create a vector of p branches $F = (f_1,\ldots,f_p)$ with a maximal length of f_{max_length} and a maximal depth of f_{max_depth}.
3. Evaluate all branches to create a vector of outputs $O = (o_1,\ldots,o_p)$ for each of the p fragments.
4. Create a set of k cut-points $M = \{m_1,\ldots,m_k\}$ randomly, where a manipulation point is a node (internal or external) of the solution candidate.
5. Initialize the set of moves $N=\{\}$.
6. If mode is semantic then continue at 7. else continue at 8.
7. For each manipulation point (semantic):
   (a) Calculate the output out_m of the branch.
   (b) Calculate the R^2 value of the output out_m of the branch and all outputs of the fragments O.
   (c) Select n fragments with the highest R^2 value (< 1) and create n moves from the solution candidate replacing the branch at the manipulation point with each of the selected fragments and add the new moves to N.
   (d) Goto 9.
8. For each manipulation point (random):
   (a) Select n fragments from the pool F randomly and create n moves from the solution candidate replacing the branch at the manipulation point with each of the selected fragments and add the new moves to N.
9. Remove all moves from N which do not fulfill the aspiration criterion and would manipulate a node which is in the tabu-list. If all moves are tabu then keep one randomly chosen move.
10. Evaluate the fitness of all moves by temporarily applying each move and evaluating the accuracy of the resulting model.
11. Select the move that leads to the best accuracy (might be smaller than the accuracy of the current solution candidate).
12. Apply the move to the solution candidate to produce the current solution of the next iteration and add the manipulated point to the tabu-list.
13. Update the best-of-run solution if the current solution candidate has a better accuracy.
14. If the stopping criterion is true then stop and return the best-of-run solution, otherwise go to step 4.

In the move generation steps the maximal depth and length of the tree is restricted by generating only moves that produce trees smaller than the given limits. In semantic mode the most similar branches are selected, however, perfectly similar replacement branches (R^2=1) are not considered because such a move would have no effect on the output of the solution.

When a move is applied to the solution the existing branch is not simple replaced with the branch from the pool, but instead a swap operation is performed. The existing branch is added to the pool instead of the just inserted branch.

5.3. Tabu Criterion

Moves that would manipulate a node that has recently been replaced in another move are tabu. The length of the tabu list determines how long a manipulated node is locked. For comparing two moves we calculate the path to the manipulated node starting from the root node. This path can be represented as a list of integers where each element is the index of the sub-tree that leads to the manipulated node. For instance in Figure 2 the path [1,0] leads to the red node which is a new node that has been inserted in a previous move. The path must match exactly with the path of a previous move to make a move tabu, i.e. changing a sub-branch below a previously inserted branch is allowed.

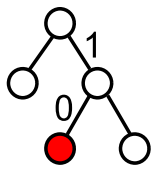

Figure 2: Example for a Path to a Manipulated Position in the Tree.

5.4. Aspiration

Several aspiration criteria are used to allow moves that would otherwise be tabu. The first aspiration criterion allows moves which would lead to a new best-of-run quality.

The second aspiration criterion is necessary to allow moves that manipulate a path that has recently been manipulated in an earlier move, when a more recent move replaced a larger branch containing the previously inserted branch. Figure 3 shows the solution

and the tabu manipulation points for three consecutive moves. First the single red node is replaced and the point is locked. In the second move the first move is undone because the whole right branch is replaced. As stated above only the manipulated point is set to tabu. The newly added nodes below the manipulation points can be changed immediately in consecutive moves. In particular the point replaced in the first move which is still in the tabu list can be manipulated again. This is accomplished through the aspiration criterion. In the third step the whole left branch is replaced. Now both, the direct left and right children of the root node are locked. All other nodes might be manipulated in consecutive moves.

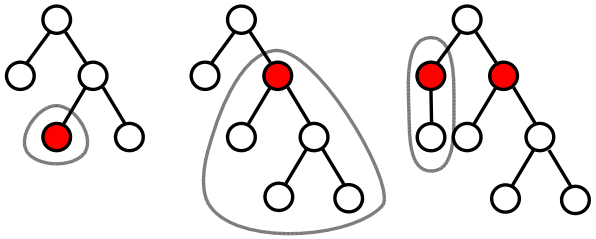

Figure 3: Aspiration Criterion for Moves That Are Undone with a Later Move.

The third aspiration criterion incorporates semantic information. The underlying idea is that moves should be aspired when the solution is changed in the same location, but in such a way that the solution is less similar to a previous solution than the current solution (i.e. semantically walking backwards). In particular, it is allowed to make moves that lead to a new solution when it is less similar to a previous solution than the current solution, even when the manipulation point would be tabu. Figure 4 shows the relevant similarities for two visited solutions (with branches *A* and *B*) and a potential new solution with a branch *C*, where *A, B* and *C* are all on the same location of the solution. First the semantic similarity of the two branches of the previous move ($R^2(A,B)$) is determined. If the semantic similarity of the original branch and the replacement branch of the current move ($R^2(A,C)$) is smaller than $R^2(A,B)$ the move is allowed. A move that would insert a branch *C* that is again more similar to the original branch *A* is not allowed.

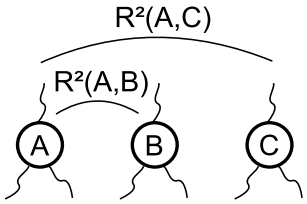

Figure 4: The semantic aspiration criterion allows tabu moves (C) if the result is less similar to the original (A) than the previously inserted branch (B).

5.5. Application of Moves

Because semantic similarity of branches is determined using the squared correlation coefficient which is invariant to changes in scale and location it is necessary to linearly scale a branch before inserting it into the solution. This has the negative effect that four additional tree nodes are necessary for each inserted branch (two constants and an addition and multiplication operator). Especially, the constant nodes are problematic because the correlation is zero for all possible replacements. This has the effect that constant nodes are replaced with a random branch from the pool which is then scaled in such a way to produce a constant output (multiplication with zero). In order to prevent such manipulations we check this special case and replace constant nodes only with unscaled branches. This is a rather uncomfortable workaround and should be addressed in a better way future work. An alternative would be to use the mean of squared errors as the metric for semantic similarity, but then the pool size should be much larger to increase the likelihood to find similar replacement branches.

Another important aspect of move application is that the replaced branch in the solution is added to the pool of replacement branches. This is necessary to prevent losing already well fitted branches because of a bad random manipulation. If the replaced branch is instead added to the pool, this branch can later be reused in a different, or even the same position in the solution.

5.6. Model Size Limits

Interestingly, without limiting the size of the model we also observed a steady growth of the model in initial experiments. This growth resulted from the fact that small fragments were replaced by on average larger fragments. Inserting a new branch instead of a terminal node leads to a set of new docking points which are definitely not in the tabu list and the algorithm can then easily find another move leading to a minor quality improvement replacing one of the newly introduced terminal nodes.

To prevent this growth of solutions we introduced limits for the maximal tree size and depth that are checked when generating possible moves.

6. DISCUSSION

A major issue in the configuration of the algorithm proofed to be finding good parameter values for the number of cut-points and the number of replacement branch for each cut-point. If the number of cut-points is small then it is very likely that a move replaces a branch near the root of the solution which leads to a deleterious move that destroys the already well fitted solution frequently. On the other hand if the number of cut-points is large then it is very easy to find a move that has only a minor positive or negative effect on the solution fitness by replacing a node deep down in the tree with a similar node. This leads to the unwanted behavior where tabu search plateaus on a sub-optimal

solution without any significant changes until a deleterious manipulation occurs.

Initial experiments showed that the performance of the algorithm depends very strongly on the balance of these two parameter values; however, we have not yet executed experiments to determine good general parameters and instead rely on a reasonable default setting.

Another difficulty for the algorithm are constants in the solution and in the branches in the replacement pool. As constants have zero correlation with all possible outputs, the semantic similarity of all branches to constants is very low. This makes it necessary to handle constants specifically in the algorithm. In the experiments we have not allowed constants in the initial branches in the replacement pool; however, through the linear scaling of branches constants are automatically introduced into the solution over the run. We leave the analysis of the effects of constants and the development of a better way of handling constants for future work.

One advantage of trajectory-based algorithms is that changes to solutions can often be calculated incrementally. For symbolic regression solutions this is principally also possible because only the parents of the newly inserted branch have to be re-evaluated. The outputs of all other branches can be cached as they are not affected by the insertion of a new branch. This would also be possible for sub-tree crossover in genetic programming; however, because tabu search only uses a single solution at any time instead of a whole population, caching becomes feasible.

The calculation of semantic similarity leads to a rather large overhead as for each move the output of the original branch has to be compared to the output of all fragments in the pool and thus the asymptotic runtime complexity is *O(kp)*. Additionally, the calculation of the squared Pearson's correlation coefficient grows linearly with the number of observations.

7. EMPIRICAL EVALUATION

For the empirical evaluation of the proposed approach we used two sets of relatively easy benchmark instances defined by Keijzer and Nguyen as described in (McDermott et al., 2012). We compared three different algorithms: symbolic regression based on genetic programming and the proposed approach using semantic move generation and random move generation. The parameter settings for genetic programming and tabu search are shown in Table 1 and Table 2. Because the benchmark instances are relatively easy we used only a small number of iterations for tabu search and the number of evaluated solutions for GP was set in such a way to allow the at least the same number of evaluated solutions as for tabu search.

Table 1: Genetic Programming Parameter Settings

Parameter	Value
Population size	100
Offspring selection	Success ratio = 1 Comparison factor = 1
Parent selection	Gender-specific (random/proportional)
Max. evaluated solutions	50,000
Max. selection pressure	100
Tree creation	PTC2
Size limits	Length=80, Depth = 12
Function set	+,*,%, variables (no random constants)

Table 2: Tabu Search Parameter Settings

Parameter	Value
Iterations	1,000
Number of cut-points (k)	1
Number of fragments (n)	100
Size limits	Length=80, Depth = 12
Fragment size limits	Length=8, Depth=4
Pool size (|F|)	1,000
Function set	+,*,%, variables (no random constants)

We executed ten independent repetitions for each algorithm configuration and benchmark instance.

8. RESULTS

The results of the empirical evaluation on the simple benchmark problems are shown in Table 3. It can be clearly seen that the genetic programming configuration works best for almost all instances and finds the solution for about half of the instances. There is no clear difference between tabu search with random move generation in comparison to tabu search with semantic move generation. In both tabu search configurations the semantic aspiration criterion has been used.

Table 3: Result Comparison

Instance	OSGP	TS random	TS semantic
Keijzer-1	**0.9552**	0.7141	0.7594
Keijzer-4	**0.1612**	0.0678	0.1183
Keijzer-5	**1.0000**	0.9931	0.9966
Keijzer-6	**1.0000**	0.4901	0.5911
Keijzer-7	**0.9999**	0.2867	0.1849
Keijzer-8	0.9732	0.9726	**0.9789**
Keijzer-10	**0.9990**	0.9768	0.9784
Keijzer-11	**0.9787**	0.9581	0.9435
Keijzer-12	**0.9995**	0.9692	0.9534
Keijzer-13	**0.8976**	0.7491	0.6763
Keijzer-14	**0.9793**	0.8000	0.8206
Keijzer-15	**0.9771**	0.9449	0.9397
Nguyen-1	**1.0000**	0.9879	0.9975
Nguyen-2	**1.0000**	0.9504	0.9929
Nguyen-3	**1.0000**	0.9903	0.9963
Nguyen-4	**1.0000**	0.9909	0.9925

Nguyen-5	**1.0000**	0.9726	0.9598
Nguyen-6	**1.0000**	0.9780	0.9522
Nguyen-7	**1.0000**	0.9863	0.9717
Nguyen-8	**0.9999**	0.9877	0.9602
Nguyen-9	**0.9993**	0.9961	0.9962
Nguyen-10	**0.9998**	0.9960	0.9962

9. DISCUSSION OF RESULTS

The result of the algorithm comparison is rather devastating and clearly indicates that the described approach for tabu search based symbolic regression is still rather immature, as it fails to find good solutions even for easy benchmark instances. Detailed analysis of the results for the Keijzer-6 and Keijzer-7 instances where TS performed a lot worse than GP showed that in these runs a scaling problem occurred. The algorithm found almost correct solutions, but they were at the bottom of a deep tree where the upper layers scaled and re-scaled the output multiple times. This has lead to numeric instabilities which then reduced the accuracy of the solutions.

Generally the solutions generated by TS often contained deeply nested structures with many fragments introduced by the linear scaling of replacement branches. In particular, solutions often contained many constants. This issue should be analyzed in more detail in future work as constants are particular bad for the semantic move generation as mentioned above.

Another aspect that should be researched in more detail is the issue that replacing a branch near the root leads to a complete disruption of the solution while replacing branches deep down in the tree often have only a minor effect on the solution output. This is a direct consequence of the tree-based representation and makes design of useful move operators very difficult. Future work should consider different move operators than just simply replacing branches as discussed in this contribution.

Finally it could be helpful to add a form of diversification strategy for instance through long-term memory. In this paper we only used a tabu-list for short term memory to facilitate exploration of the fitness landscape in addition to the local improvement of the semantic move operator.

ACKNOWLEDGMENTS

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

REFERENCES

Abdel-Rahman H., Emad Mabrouk, Masao Fukushima, 2011, Tabu Programming: a New Problem Solver through Adaptive Memory Programming over Tree Data Structures, *International Journal of Information Technology and Decision Making* 10(2):373 – 406.

Balicki, J., 2009, Some numerical experiments on multi-criterion tabu programming for finding Pareto-optimal solutions, *WSEAS Transactions on Systems* 8(1):241 – 250.

Balicki, J., 2007, Hierarchical Tabu Programming for Finding the Underwater Vehicle Trajectory, *International Journal of Computer Science and Network Security* 8(11):32 – 37.

Glover, F., Laguna, M., 1999, *Tabu Search*, Kluwer Academic Publishers.

Hastie, T., Tibshirani, R., Friedman, J., 2009, *The Elements of Statistical Learning*, Springer.

Krawiec, K., 2011, Learnable Embeddings of Program Spaces, *Proceedings of the 14th European Conference on Genetic Programming*, LNCS (6621) 166 – 177, Springer Verlag.

Krawiec, K., 2012, Medial Crossovers for Genetic Programming, *Proceedings of the 15th European Conference on Genetic Programming*, LNCS (7244) 61 – 72, Springer Verlag.

Koza, J., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press

McConaghy, T., 2011, Fast, Scalable, Deterministic Symbolic Regression Technology, *Genetic Programming Theory and Practice IX*, Springer

McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., O'Reilly, U.-M., Genetic Programming Needs Better Benchmarks, *Proceedings of GECCO'2012*, July 7-11, 2012, Philadelphia, Pennsylvania, pp. 791 – 798, ACM.

Nguyen, Q.U., Nguyen, X.H., O'Neill, M., 2011, Examining the landscape of semantic similarity based mutation, *Proceedings of the 13th annual conference on genetic and evolutionary computation (GECCO'11)*, ACM, 12-16 July 2011, pp. 1363 – 1370.

Nguyen, Q.U., Nguyen, T.H., Nguyen, X.H., O'Neill, M., 2010, Improving the Generalisation Ability of Genetic Programming with Semantic Similarity based Crossover, *Proceedings of the 13th European Conference on Genetic Programming (EuroGP2010)*, LNCS (6021) 184 – 195, Springer