

# Genetic Programming for Proactive Aggregation Protocols

Thomas Weise, Kurt Geihs, and Philipp A. Baer

University of Kassel, Wilhelmshöher Allee 73, D-34121 Kassel, Germany  
{weise, geihs, baer}@vs.uni-kassel.de  
<http://www.vs.uni-kassel.de>

## Preview

This document is a preview version  
and not necessarily identical with  
the original.

<http://www.it-weise.de/>

**Abstract.** We present an approach for automated generation of proactive aggregation protocols using Genetic Programming. First a short introduction into aggregation and proactive protocols is given. We then show how proactive aggregation protocols can be specified abstractly. To be able to use Genetic Programming to derive such protocol specifications, we describe a simulation based fitness assignment method. We have applied our approach successfully to the derivation of aggregation protocols. Experimental results are presented that were obtained using our own Distributed Genetic Programming Framework. The results are very encouraging and demonstrate clearly the utility of our approach.

## 1 Introduction

Determine the highest temperature measured by a sensor in a given area. Find the average load of all computers in a grid. Aggregation functions with their ability to summarize information in a certain, user-specified way are a very important building block for distributed applications [1]. As a standard service of databases, SQL-queries allow the user to aggregate locally available data in one or multiple tables. Performing aggregation in a sensor network [2], as done in the introductory examples, however is more complicated. Only data obtained from the local sensors is available on a node. In order to determine the desired aggregate, a node needs to exchange messages with other nodes in its neighborhood. Therefore, reactive and proactive protocols can be distinguished. Reactive protocols are used to compute queries issued by a single node and offer the result of the query to this node only. Proactive protocols update aggregation values continuously and make them globally available.

In this paper we demonstrate that Genetic Programming is an effective technique to derive proactive aggregation protocols. We describe how such protocols can be specified in an abstract manner and introduce a simulation-based fitness assignment method used to evaluate these specifications.

## 2 Related Work

In our work we strongly refer to Jelasyty et al. who have defined and evaluated many proactive aggregation protocols for large-scale overlay networks [3]. Although their communication model is gossip-based, our abstract protocol specifications will work with epidemic [4] or SPIN-based [5] communication as well.

Protocol generation has been a field of application for Genetic Algorithms in the past ten years. Genetic Algorithms have been used to optimize different aspects of protocols like communication, implementation costs, and performance [6–9]. While these approaches separate the application logic itself from the protocol generation and focus on the communication, we introduced in our previous work [10] a versatile Genetic Programming technique for simple distributed algorithms combining both of these aspects. In this paper we shift the emphasis towards the application logic (i.e. the formulas needed to compute aggregation values) while using an implicit communication pattern.

As we will show in the next sections, this mathematical application logic is based on a form of symbolic regression [11, 12] but exceeds its scope in many ways.

## 3 Basic Model and Protocol Specification

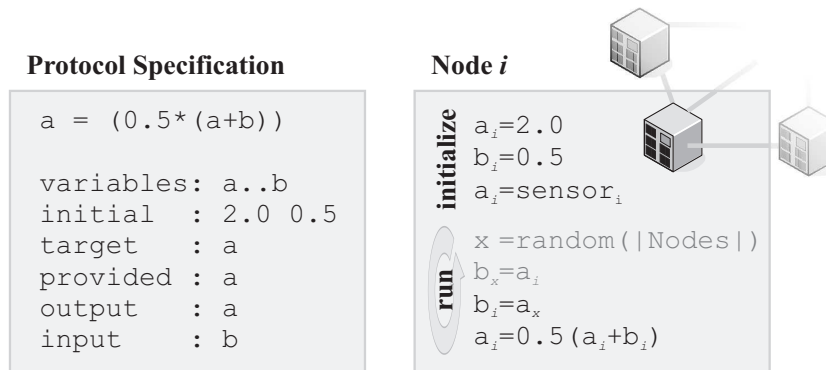
For our studies we use a system model similar to the one introduced by Jelasyty et al. [3]. A distributed system is regarded as a large collection of nodes that communicate through message exchange. These nodes are assumed to be connected by a routed network allowing all nodes to exchange messages with each other.

In order to compute an aggregate value (see *target* in Figure 1), each node owns a local storage, consisting of  $n$  variables  $v_i : i \in 1..n$  named  $a, b, c, \dots$  and so on. It furthermore knows one value, for example a sensor measurement, needed to compute the aggregate. With this *provided* value,  $0 < m \leq n$  of the variables will be initialized while the others have constant numerical *initial* values.

To perform the aggregation, data exchange and variable updates are needed to be carried out iteratively in a loop. The data exchange in the network is performed as follows:  $0 < p < n$  of the variables are marked as *output* and also  $p$  variables are marked as *input*. At the beginning of each iteration step, a partner node  $y$  is selected for each node  $x$  in the network. This could, for example, be a node in transmission range chosen from the direct neighborhood. The values of the output variables of node  $x$  will then be stored in the input variables of node  $y$ , and vice versa. After this is done, the variables are updated by computing the formulas  $f_j : j = 1..q$  on every node, where each formula  $f_j$  assigns a new

value to one of the variables  $v_i$ . Formulas are expressions built with operators such as  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$  (power),  $|*|$  (absolute value),  $min$ ,  $max$ , constants, and with variables.

Figure 1 displays such a protocol specification, able to compute the aggregate *Average*. This optimal protocol, which was found using our own Distributed Genetic Programming Framework [13–15], is exactly the same as the solution proposed by Jelasity et al. [3].



**Fig. 1.** A protocol specification for the distributed average, found using GP.

The algorithms in this format can be regarded as platform-independent models. They can easily be applied to any given target platform, either by implementing them automatically or by hand. The target platform then determines the underlying transmission protocol, the programming language of choice, and the intervals in which the data exchange specified by the protocol will occur.

## 4 The Fitness Assignment Method

In principle, the evolution of protocol specifications is a complex extension of symbolic regression [11]. Instead of trying to evaluate a single formula with a single input parameter one time for each test set [12], multiple formulas with multiple inputs are evaluated  $T$  times in a loop on multiple nodes in a simulated network.

When performing symbolic regression, the functionality of a formula can be determined by computing its result for a number of test sets. The difference between the values computed and the expected results then clearly indicates the fitness.

In case of engineering an aggregation protocol  $P$ , each test set is an  $n$ -dimensional vector where  $n$  is the number of simulated nodes. Running the protocol yields an aggregate value  $a_y$  for each one of the nodes ( $y = 1..n$ ) in a network. To decrease the inaccuracy as well as to reward protocols with at

least one good approximation, we compute the sum  $\delta$  of the average and the maximum deviation of these aggregates from the correct result  $A$ . As shown in equation 1,  $\delta_x(t)$  is the deviation sum of the aggregate values  $a_y$  computed by the nodes  $y$  for a single test set  $x$ . This value should be minimized.

$$\delta_x(t) = \max_{\forall \text{ nodes } y} \{|a_y(t) - A_x|\} + \text{avg}_{\forall \text{ nodes } y} \{|a_y(t) - A_x|\} \quad (1)$$

Instead of calculating this value at the end of each simulation, we integrate it for each of the  $T$  iterations of the protocols loop (equation 2). Therefore we do not only measure protocol accuracy but also convergence speed: If a protocol converges faster than another one of the same accuracy, the deviation sums will begin to shrink sooner.

This approach has a minor drawback: Protocols which simply “guess” a constant value for the aggregates may be rewarded with a better fitness (smaller  $\delta$  values) in most of the iteration steps – if they guess well. Weighting the last iteration with the square root of the total iteration count  $T$  has proven to be a useful countermeasure to prevent this phenomenon.

The formula for the functional fitness  $f$  of the protocol  $P$  computed using all test sets  $x$  is presented in equation 2. This value is subject to minimization.

$$f(P) = \sum_{\forall \text{ testsets } x} \frac{1}{|A_x|} \frac{1}{|T + \sqrt{T}|} \left[ \sum_1^{T-1} \delta_x(t) + \sqrt{T} \delta_x(T) \right] \quad (2)$$

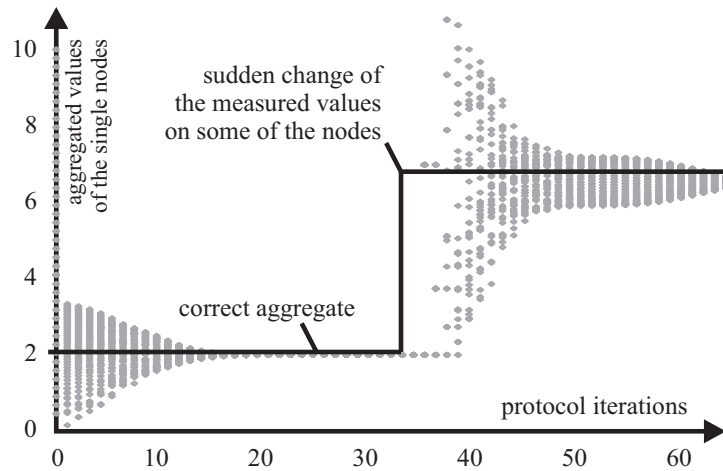
Other quality aspects of an aggregation protocol are the size of the messages exchanged, the number of variables needed and the complexity of the formula expressions. Introducing a second fitness function, we use a weighted multi-objective approach to optimize these aspects too.

## 5 Example

Trivial basic aggregation functions like *Minimum*, *Maximum* and *Average* can be evolved by using the fitness assignment method of the previous section within only a few generations using Genetic Algorithms with less than 2000 individuals. To prove that our approach is also suitable for more complex aggregates, we chose the following function to be approximated: Each node owns one sensor producing one measured value. We are interested in the average of the square roots of the absolute values of these measurements.

The most trivial solution would be that the nodes compute initially the square roots of their sensor measures and then execute the *Average* protocol of figure 1. In a real application, the sensor measurements will change over time forcing the protocol to incorporate new values (see figure 2). This is not possible with the trivial solution discussed, which therefore is not a valid protocol specification.

Since the search algorithm implementations of the Distributed Genetic Programming Framework, which we use for our studies, are intended for maximization only, the functional fitness (see equation 2) was inverted. A Genetic



**Fig. 2.** A Protocol sensitive to input data change.

Algorithm with a bigger population of 8192 individuals was applied, yielding the solution depicted in figure 3. This figure shows the functionally best pareto-optimal protocol, which produces approximation values differing only by a very small error from the correct aggregate. The diagram in the right part of figure 3 shows the convergence of the protocol. 50 nodes were initialized with values uniformly distributed over the interval  $[-1, 9]$  and quickly converged to a constant shared by all nodes, only deviating less than 1% from the real solution. These deviations rise considerably if the measured input values change, as mentioned at the beginning of this section. The protocol is however able to adapt to bigger changes – it is also the source of figure 2.

#### Protocol Specification

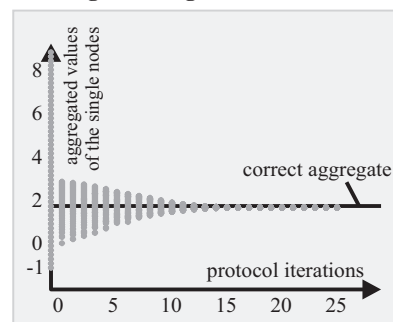
```

b = (b/a)
a = |c|
c = max((a*b), (a-(b*a)))
a = ((a*b)^0.5000000022)

variables: a..c
initial  : 3.14 0.11 154
target   : a
provided : c
output   : a
input    : b

```

#### Convergence Diagram

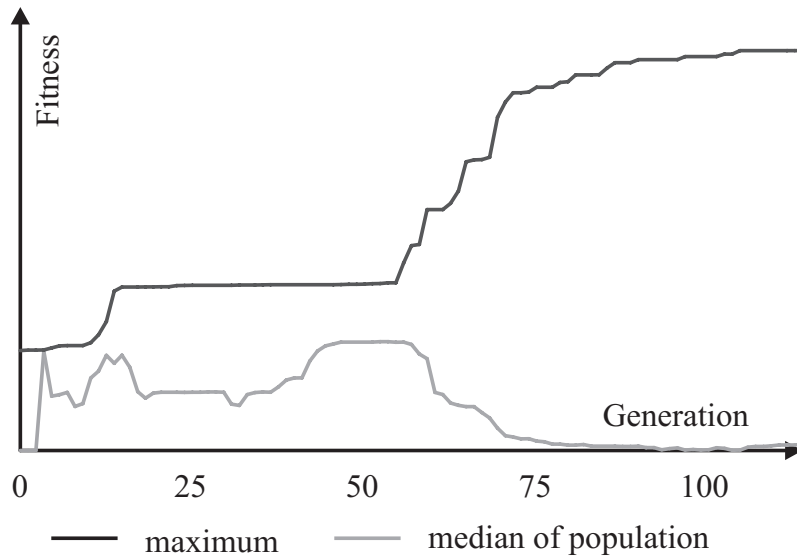


**Fig. 3.** A protocol evolved for the example problem and its convergence behavior.

It is important to mention here that genetically evolved protocol specifications have a strong affinity to memorize test sets. Symbolic regression often does not yield the “original” formula which was used to create the test sets but a function which only resembles this formula closely. In symbolic regression this effect is wanted in many cases – for protocol generation it is devastating. Sometimes generated protocols which seem to be excellent solutions for a given aggregation problem turn out to be disguised decision tables. In our work we use two simple means to circumvent this problem:

1. We use many test sets (empirically determined  $\geq 240$ ).
2. The values contained in the test sets differ in scale, sign, and generating distribution function (uniform, normal, many zeros/few ones).

Another interesting phenomenon concerns the fitness of the protocol specifications. Plotting the maximum and the median of the functional fitness of each generation exhibits an interesting behavior (see figure 4): while the fitness of the best individuals becomes significantly better step by step, the fitness of most of the population degenerates at the same rate. A possible explanation for this phenomenon would be that the more accurate protocol specifications get, the more destruction can be done by genetic operators. This effect is increased by the non-functional fitness function applied which adds pressure towards compact protocol design – and therefore tends to shrink those parts of the specification that have smaller impact on the total results.



**Fig. 4.** The maximum and median fitness of the whole population of the Genetic Algorithm for the average-of-square-roots problem, plotted against the generations.

## 6 Conclusion and Future Work

In our research we were able to demonstrate the utility of Genetic Programming for deriving proactive aggregation protocols. Thus, after having shown its usefulness for breeding assembler-like distributed algorithms in [14], in this paper we have presented our second successful application of GP to create emergent phenomena: A global property of a whole (network) has been transformed into behavior rules for an individual (node). In our future work we will go on evaluating different fields of application of GP as a means for creating distributed algorithms. Our DGPF [13] will be improved further with the ultimate goal to provide a comprehensive platform for automated software creation for sensor networks.

## References

1. Robbert van Renesse. The importance of aggregation. (2584):87–92, 2003.
2. Chee-Yee Chong and S.P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug 2003.
3. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(1):219–252, 2005.
4. Márk Jelasity and Alberto Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 102–109, Tokyo, Japan, Mar 2004. IEEE Computer Society.
5. Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM Press.
6. K. El-Fakih, H. Yamaguchi, G. Bochmann, and T. Higashino. A method and a genetic algorithm for deriving protocols for distributed applications with minimum communication cost. In *Proceedings of Eleventh IASTED International Conference on Parallel and Distributed Computing and Systems*, Nov 1999.
7. Lidia Yamamoto and Christian Tschudin. Genetic evolution of protocol implementations and configurations. In *IFIP/IEEE International workshop on Self-Managed Systems and Services (SelfMan 2005)*, 2005.
8. F. Comellas and G. Giménez. Genetic programming to design communication algorithms for parallel architectures. *Parallel Processing Letters*, 8(4):549–560, 1998.
9. Marcio Nunes de Miranda, Ricardo N. B. Lima, Aloysio C. P. Pedroza, and Antonio C. de Mesquita. HW/SW codesign of protocols based on performance optimization using genetic algorithms. Technical report.
10. Thomas Weise and Kurt Geihs. DGPF - an adaptable framework for distributed multi-objective search algorithms applied to the genetic programming of sensor networks. In Jurij Šilc Bogdan Filipič, editor, *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Application, BIOMA 2006*, International Conference on Bioinspired Optimization Methods and their Application (BIOMA), pages 157–166. Jožef Stefan Institute, Ljubljana, Slovenia, Oct 2006.

11. John R. Koza. *Genetic Programming, On the Programming of Computers by Means of Natural Selection*. A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1992 first edition, 1993 second edition, 1992.
12. Gunther R. Raidl. A hybrid GP approach for numerically robust symbolic regression. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 323–328, University of Wisconsin, Madison, Wisconsin, USA, 22–25 1998. Morgan Kaufmann.
13. Distributed Genetic Programming Framework. SourceForge project, see <http://sourceforge.net/projects/DGPF> and <http://DGPF.sourceforge.net/>.
14. Kurt Geihs Thomas Weise. Genetic programming techniques for sensor networks. In *Proceedings of 5. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, pages 21–25, Jul 2006.
15. Thomas Weise. Genetic programming for sensor networks. Technical report, Jan 2006.

```
@inproceedings{WGB2007DGPFb,  
  author      = {Thomas Weise and Kurt Geihs and Philipp Andreas Baer},  
  title       = {{Genetic Programming for Proactive Aggregation Protocols}},  
  doi         = {10.1007/978-3-540-71618-1},  
  pages       = {167--173},  
  affiliation = {University of Kassel, FB-16, Distributed Systems Group,  
                Wilhelmsh{"o}her Allee 73, 34121 Kassel, Germany},  
  keywords    = {Genetic Programming, Proactive Aggregation Protocols,  
                Aggregation, Sensor Networks, DGPF, Symbolic Regression},  
  booktitle   = {Proceedings of Adaptive and Natural Computing Algorithms,  
                8th International Conference, ICANNGA 2007, Part II},  
  location    = {Warsaw University of Technology, Warsaw, Poland},  
  month       = apr # {~11--14,},  
  ISSN        = {0302-9743},  
  publisher   = {Springer Berlin Heidelberg New York},  
  series      = {Lecture Notes in Computer Science (LNCS)},  
  volume      = {4432/2007},  
  year        = {2007},  
  isbn        = {978-3-540-71590-0},  
  url         = {http://www.it-weise.de/documents/files/W2007DGPFb.pdf},  
}
```