Genetic Programming Techniques for Sensor Networks

Thomas Weise, Kurt Geihs

University of Kassel, Wilhelmshöher Allee 73, Kassel, Germany
{weise|geihs}@vs.uni-kassel.de

Abstract. In this paper we present an approach to automated program code generation for sensor nodes and other small devices. Using Genetic Programming, we are able to discover algorithms that solve certain problems. Furthermore, non-functional properties like code size, memory usage, and communication frequency can be optimized using multiobjective search techniques. The evolution of algorithms requires program testing, which we perform using a customized simulation environment for sensor networks. The simulation model takes into account characteristic features of sensor nodes, such as unreliable communication and resource constraints. An application example is presented that demonstrates the feasibility of our approach and its potential to create robust and adaptive code for sensor network applications.

1 Introduction

Today we experience a growing demand for distributed systems consisting of sensor nodes [1]. As explained in [2], such devices are restricted in resources like memory size, processing speed, and battery power. The communication among them is costly in terms of energy and not reliable either. Furthermore, the topology of sensor networks is volatile and usually cannot be determined a priori, enforcing self organization. Algorithms and protocols normally applied to distributed systems are therefore often insufficient and need to be replaced with specialized counterparts.

These requirements make programming sensor nodes a demanding task for software developers because the design of sensor network applications must pay attention to aspects that are not directly related to the application functionality itself. We claim that Genetic Programming techniques are an effective means to automatically discover powerful programming solutions for sensor networks. Clearly, Genetic Programming is not suited to produce very large programs for general application tasks. However, we view sensor nodes as ideal targets for Genetic Programming since these nodes can only perform a limited functionality due to their resource constraints.

In this paper we introduce DGPF, the Distributed Genetic Programming Framework [3], which allows us to automatically discover distributed algorithms for given problems. Additionally, such algorithms can be optimized in various ways, taking energy consumption into account as well as memory usage or code size. In the next section we describe how Genetic Programming can be applied to sensor networks followed by an example in the third section. At the end of the paper, a short

summary of related work is given along with future work and a conclusion.

2 Genetic Programming and Sensor Networks

For a long time, Genetic Algorithms have been used in science to derive solutions for any type of problems, from construction of wind turbines to pattern-recognition systems. The application of Genetic Algorithms with the goal to evolve computer programs is called Genetic Programming. This section will give a brief overview on how Genetic Algorithms work in common and how they can be applied to sensor networks.

2.1 Genetic Algorithms

As shown in Fig. 1, Genetic Algorithms start with an initial population of random solution candidates called individuals. In our case, the individuals are small programs that can be executed on sensor nodes. As in nature, the population will be refined step by step in a cycle of computing the fitness of its individuals, selecting the best individuals and creating a new generation derived from these. If a reasonable good solution has evolved, the algorithm will terminate.

Randomized optimization algorithms are called "multiobjective" if they permit the specification of more than one optimization objective. Using our DGPF, several fitness functions can be defined, allowing us to optimize programs not only for functionality but also for nonfunctional requirements like energy consumption and communication frequency. Furthermore, different search algorithms like randomized Hill Climbing and Genetic Algorithms can be combined to speed up the optimization process.

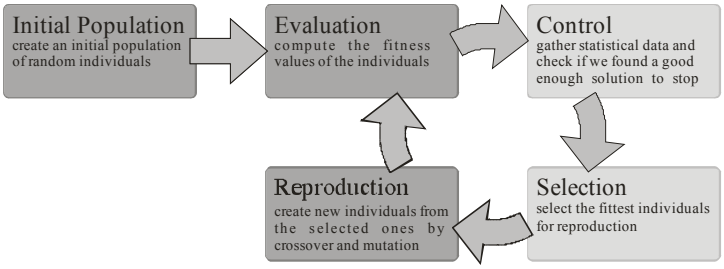

Fig. 1. Common cycle of Genetic Algorithms.

In our case the individuals that are being evolved are algorithms in the form of small programs. The functionality and effectiveness of such an algorithm can be determined by simulating it on a virtual hardware representing a single sensor node. In terms of distributed algorithms, it is not sufficient to simulate only one sensor node. Thus multiple instances of the program will be executed in a network simulator in parallel. The following two subsections describe the virtual hardware and the network model used in our simulations.

2.2 Virtual Hardware

A sensor node is modeled as an automaton that consists of a virtual hardware holding its execution state and a program running on that hardware. Unlike most other approaches in Genetic Programming which grow stateless functions, we have developed an architecture with a fixed-sized memory. The instruction set can be reduced to the one introduced by Teller [4], granting the Turing completeness needed to model real distributed algorithms or network protocols. Like real microprocessors, direct and indirect memory access and a collection of arithmetic expressions are included in the instruction set as well as conditional jumps. Communication is also modeled with primitive directives which allow storing memory words in an output buffer and transmitting the buffered data. A single message can be received into the input buffer and will be processed by reading the received words sequentially. Newly incoming messages get lost if the input buffer is already occupied.
The example in the next section displays some of the available instructions (see Fig. 2).

2.3 Simulation

The network simulation provides additional statistical data for each automaton, holding information on the number of messages sent, lost, and successfully processed.

As in real networks, many automata run asynchronously at approximately the same speed which, however, might differ from node to node and cannot be regarded as constant either. To grant realistic simulations, the network model has the following properties:

1. The links between the nodes are randomly created, yet it will be ensured that there are no network partitions.
2. Messages are simple word sequences with no predefined structure.
3. Messages cannot be sent directly. Like radio broadcasts they will be received by any node in transmission distance. Finding out which message is of concern will be in the responsibility of each node.
4. Messages can get lost without special cause.
5. Transmissions may take a random time until they reach their target.
6. The collision of two transmissions underway leads to the loss of both messages.

3 Example Algorithm

One example problem that we have solved with the DGPF is the so-called election problem. This problem is well known in the area of distributed computing and therefore we have used it to validate our approach. Election means to select one node out of a group of nodes, for instance to act as a communication relay.

Each node owns a unique identifier and all nodes must know the ID of the elected node after the election algorithm has terminated. One method to perform such an

election would be to select the node with the maximum ID. We therefore introduce a functional fitness function that evaluates how many nodes know the maximum ID after a given amount of time. Additionally, we enter three non-functional fitness functions into the evolution: parsimony pressures for minimum code size, minimum memory size and a minimum transmission count. Each automaton is initialized with its own ID in its memory cell.

It took several hours to obtain the solution depicted in Fig. 2 using an older version of our framework on a 3 GHz Pentium 4 PC. Most time was spent on the network simulation. Due to many optimizations, the current version of the DGPF runs significantly faster and also incorporates various new distribution schemes [3] for Genetic Algorithms resulting in further speedup, depending on the number of available computers.

The discovered algorithm seems to be simple and quite efficient for the problem definition: The nodes initially send the contents of their first memory cell (their ID) to all neighbors. If and only if a node then receives a greater ID, it stores it in the first memory cell and starts the cycle again. Otherwise, no message is sent which pays respect to the parsimony pressure for minimum transmission count. The node waits instead for incoming transmissions.

```
@0:
  Send mem[0]
@1:
  mem[1]=Receive
  If(mem[1]<=mem[0]) Goto @1
@2:
  mem[0]=mem[1]
  Goto @0
```

Fig. 2. The genetically evolved election algorithm.

4 Related Work

Although Genetic Programming was invented almost 20 years ago by Koza [5] , it is a novel idea to employ it for finding distributed algorithms, in particular for sensor networks. In protocol engineering, Genetic Protocol design has already proven to be a useful and promising technology in several independent research projects [6], [7], [8]. In contrast to these approaches, our framework is not just bound to communication protocol synthesis but is able to evolve functional programs together with a dedicated protocol.

The idea of multiobjective optimization using Genetic Algorithms reaches back to the 1960s but is generally contributed to Schaffner. His VEGA-algorithm [9], although not very efficient, was the first real evolutionary algorithm which considered multiple optimization criteria in a non-trivial manner. A few years later, Zitzler et. al. showed that their advanced multiobjective evolutionary algorithm SPEA2 outperforms singleobjective GA in code size reduction by far [10]. Extending this idea, we optimize not only for code size, but also for other non-functional criteria.

5 Future Work and Conclusion

In the near future, a graphical user interface will be provided to ease the work with the DGP Framework. The quality of the results and the speed of the evolutionary process will be increased by integrating additional optimization techniques. In terms of sensor network programming, we will implement a port to the MSP430 instruction set. This will allow us to test the evolved programs on real sensor nodes like the ESBs of the ScatterWeb platform [11].

In this paper we have presented a method and a framework for automated creation of efficient algorithms for sensor networks based on Genetic Programming. We have shown the viability of the approach for a simple example. Our new framework and all results are provided as open source to the research community under the LGPL license. More information on our research as well as the fully documented Java source code of the DGPF can be found at http://dgpf.sourceforge.net.

6 References

[1] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, Michael Rohs: "Disappearing Computers Everywhere - Living in a World of Smart Everyday Objects", New Media, Technology and Everyday Life in Europe Conference, 2003, London, UK

[2] Holger Karl, Andreas Willig: "A Short Survey Of Wireless Sensor Networks", Technical Report, Telecommunication Networks Group, Technische Universität Berlin, 2003

[3] Thomas Weise, Kurt Geihs: “DGPF - An Adaptable Framework for Distributed Multi-Objective Search Algorithms Applied to the Genetic Programming of Sensor Networks”, submitted to BIOMA 2006, The 2nd International Conference on Bioinspired Optimization Methods and their Applications 9 - 10 October 2006, Ljubljana, Slovenia

[4] Astro Teller, "Turing completeness in the language of genetic programming with indexed memory", Proceedings of the 1994 {IEEE} World Congress on Computational Intelligence Volume 1, IEEE Press, 1994

[5] John R Koza: “Non-Linear Genetic Algorithms for Solving Problems”. United States Patent 4,935,877. Filed May 20, 1988. Issued June 19, 1990.

[6] Khaled El-Fakihy, Hirozumi Yamaguchiz, Gregor v. Bochmann: “A Method and a Genetic Algorithm for Deriving Protocols for Distributed Applications with Minimum Communication Cost”, Proceedings of Eleventh IASTED International Conference on Parallel and Distributed Computing and Systems, November 3-6, 1999, Boston, USA

[7] Lidia Yamamoto, Christian Tschudin, "Genetic Evolution of Protocol Implementations and Configurations", IFIP/IEEE International workshop on Self-Managed Systems and Services (SelfMan 2005), Nice, France

[8] F. Comellas, G. Gimenez: “Genetic Programming to Design Communication Algorithms for Parallel Architectures", in Parallel Processing Letters, 1998

[9] J. David Schaffner: “Multiple objective optimization with vector evaluated genetic algorithms”, in Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp.31-100, Lawrence Erlbaum, 1985

[10] Bleuler, S., Brack, M., Thiele, L. Zitzler, E.: "Multiobjecive Genetic Programming: Reducing Bloat using SPEA2", In Proceedings of the 2001 Congress on Evolutionary Computation CEC2001. IEEE Press (2001) 536-543

[11] ScatterWeb, hard- and software platform for sensor networks, including Embedded Sensor Board with TI MSP430 controller, see http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/