Multi-objective Genetic Programming for Improving the Performance of TCP

Cyril Fillon and Alberto Bartoli

University of Trieste, Via Valerio, 10, 34127 Trieste, Italy
{cfillon,bartolia}@univ.trieste.it

Abstract. TCP is one of the fundamental components of the Internet. The performance of TCP is heavily dependent on the quality of its *round-trip time (RTT) estimator*, i.e. the formula that predicts dynamically the delay experienced by packets along a network connection. In this paper we apply *multi-objective genetic programming* for constructing an RTT estimator. We used two different approaches for multi-objective optimization and a collection of real traces collected at the mail server of our University. The solutions that we found outperform the RTT estimator currently used by all TCP implementations. This result could lead to several applications of genetic programming in the networking field.

1 Introduction

Genetic programming is a powerful framework for coping with problems in which finding a solution is difficult but evaluating the performance of a candidate solution is reasonably simple [12,5]. Many engineering problems exhibit this feature and may thus greatly benefit from genetic programming techniques. Real-world engineering problems, on the other hand, can only be solved based on a trade-off amongst multiple and often conflicting performance objectives. Several approaches for such *multi-objective optimization* problems have been proposed in evolutionary computing [7], mostly for genetic algorithms [10,16,15] and more recently also for genetic programming [9,14].

In this paper we apply two techniques for multi-objective optimization in genetic programming to an important real-world problem in the Internet domain: the construction of a *round-trip time (RTT) estimator* for TCP [11,4,13]. TCP (Transmission Control Protocol) is a fundamental component of the Internet as it constitutes the basis for many applications of utmost importance, including the World Wide Web and the e-mail just to mention only the most widely known. The TCP implementation internally maintains a dynamic estimator of the round-trip time, i.e., the time it takes for a packet to reach the other endpoint of a connection and to come back. This component has a crucial importance on the overall TCP performance [11,4]. The construction of an RTT estimator is a particularly challenging problem for genetic programming because, as we shall see in more detail, an RTT estimator must satisfy two conflicting requirements, there are many solutions that are optimal for only one of the two requirements and any such solution performs poorly for the other one.

M. Ebner et al. (Eds.): EuroGP 2007, LNCS 4445, pp. 170–180, 2007.

We construct RTT estimators via multi-objective genetic programming and evaluate their performance on real traces collected at the mail server of our University. The formulas that we found outperform the RTT estimator used in all existing TCP implementations [11] — including those in Windows 2000/XP, Linux, Solaris and so on. We believe this result is particularly significant and could lead to several interesting developments and applications of genetic programming in the networking field.

The outline of the paper is as follows. The next section presents in a first part the RTT estimation problem in detail and introduces the fundamental concepts and techniques in multi-objectives optimization in a second part. Section 3 describes several multi-objective strategies used on our problem. Section 4 presents the experimental procedure used to discover new formulas which estimate RTT. Section 5 discusses the behavior of the different multi-objectives policies as well as the performances of the formula found by Genetic Programming. Finally conclusions are drawn.

2 RTT Estimation Problem

The Transmission Control Protocol (TCP) provides a transport layer, base of many other protocols used in the most common Internet applications, like HTTP (i.e., the Web), FTP (files transfer) and SMTP/POP3 (e-mail). TCP was defined in [1,2]. We provide in the following only the necessary background for this work. More details on the TCP implementation can be found in many places, for example, in [3].

TCP provides applications with a *reliable* and *connection-oriented* service. This means that two remote applications can establish a connection between

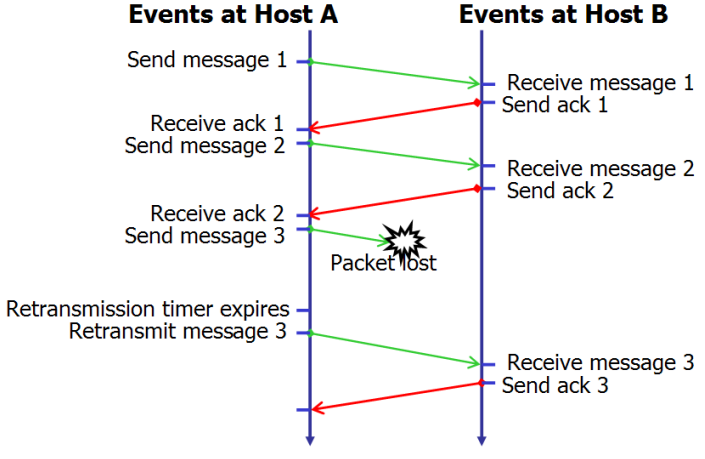

Fig. 1. An example of retransmission for the case (i)

them and that bytes inserted at one end will reach the other end reliably, that is, without losses, in order and without duplicates.

To ensure reliable delivery of packets in spite of packet losses, that may occur at lower levels of the Internet protocol stack, the TCP implementation employs internally a retransmission scheme based on *acknowledgments* as follows. Consider a connection between hosts A and B. Whenever either of them, say A, sends a packet S to the other, it sets a *retransmission timeout (RTO)*. Whenever B receives a packet S, it responds with another packet $ack(S)$ for notifying the other end that S has been indeed received. If A does not receive $ack(S)$ before RTO expires, then A resends S. Note that when RTO expires only one of the following is true, but A cannot tell which one: (i) S has been lost; (ii) S was received by B but $ack(S)$ is lost; (iii) neither S nor $ack(S)$ was lost and RTO expired too early. The fact that A resends S whenever RTO expires means that the TCP implementation *assumes* that case (i) always holds. The three cases described above are illustrated in Figure 1 and 2.

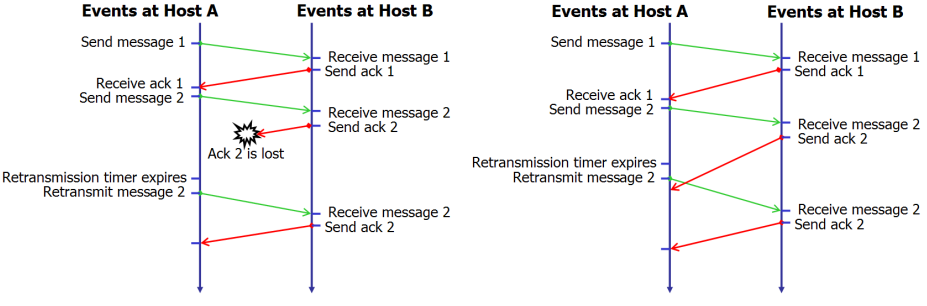

Fig. 2. Examples of retransmission for case (ii) and (iii)

Each TCP implementation selects RTO on a per-connection basis based on a formula that depends on the *round-trip time (RTT)* time, i.e., the time elapsed between the sending of a packet S and the receiving of the corresponding acknowledgment $ack(S)$. RTO should be larger than RTT to not incur in case (iii) above too often, which would waste resources at the two endpoints and within the network. On the other hand, RTO should not be much larger than RTT, otherwise it would take an excessively long time to react to case (i) which would result in a high latency at the two connection endpoints.

RTT varies dynamically, due to the varying delays experienced by packets along the route to their destination. Moreover, when sending packet S_i the corresponding RTT value $measuredRTT_i$ is not yet known. The TCP implementation thus maintains dynamically, on a per-connection basis, an *estimated RTT* and selects RTO for S_i based on the current value for $estimatedRTT_i$. This component of TCP has a crucial importance on performance of TCP [4].

Virtually *all* the TCP implementations – including those in Linux, Windows 2000/XP, Solaris and so on – maintain $estimatedRTT_i$ according to an algorithm

due to Jacobson [11]. This algorithm constructs $estimatedRTT_i$ based on the previous estimate $estimatedRTT_{i-1}$ and the previous value actually observed $measuredRTT_{i-1}$:

$$estimatedRTT_i = (1 - k_1)\, estimatedRTT_{i-1} + k_1\, measuredRTT_{i-1} \qquad (1)$$

Constant k_1 is set to $\frac{1}{8}$ allowing an efficient implementation using fixed-point arithmetic and bit shifting. Initially, $estimatedRTT$ is set to the first available $measuredRTT$. Another component of the Jacobson algorithm, not shown here for space constraints, constructs RTO_i based on $estimatedRTT_i$.

In this work we are concerned with RTT estimation only, i.e., we seek for methods for estimating RTT that are different from formula 1 and hopefully better. The construction of $estimatedRTT$ has two conflicting objectives to optimize. One would like to minimize the number of underestimates (which may cause premature timeout expiration) while at the same time minimizing the average error (which may cause excessive delay when reacting to a packet loss). The problem is challenging because optimizing the former objective is very simple — any very large estimation would work fine — but many excellent solutions from that point of view are very poor from the point of view of the average error.

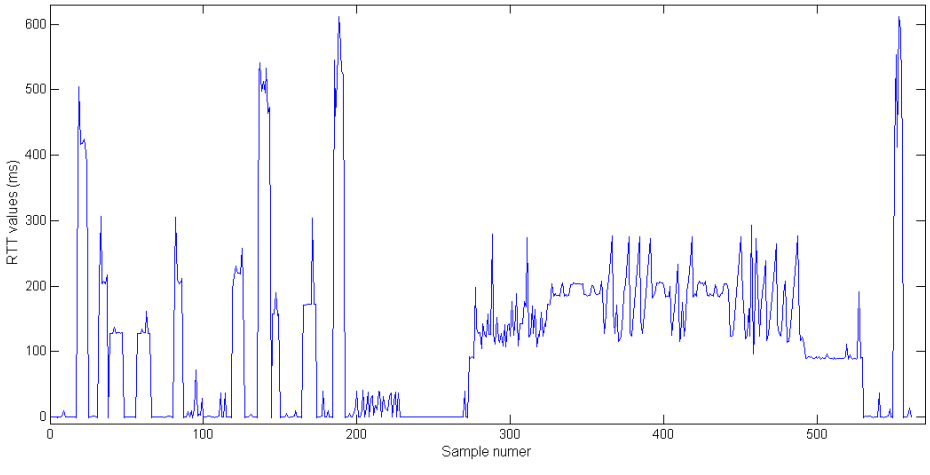

Fig. 3. Sample of RTT values for consecutive connections

An example of the sequence of RTT values measured in TCP connections is given in Figure 3 above. It is easy to realize that predicting the next value of RTT based on the past measurements, with a small error and few underestimates, is hard. A more complete characterization of real RTT traces can be found, for example, in [3].

3 Multi-objective Approaches for RTT Estimation

3.1 Multi-objective Optimization

We denote by Ω the space containing all candidate solutions to the given problem, RTT estimation in our case. In a *multi-objective optimization problem (MOP)* we want to find a candidate solution $\overrightarrow{x} \in \Omega$ which optimizes a vector of k *objective functions*: $\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x})]$. The very same nature of a MOP implies that there may be many points in Ω representing practically acceptable solutions. It is often the case that the objective functions conflict with each other [8], e.g., a solution $\overrightarrow{x}$ could be better than another solution $\overrightarrow{y}$ for some of the k objectives while the reverse could be true for the remaining objectives.

An important definition for reasoning about solutions of a MOP is the *Pareto dominance* relationship:

Definition 1 (Pareto dominance). *A solution $\overrightarrow{u} \in \Omega$ is said to dominate $\overrightarrow{v} \in \Omega$ if and only if:*

$$\forall i \in \{1, 2, \ldots, k\},\ f_i(\overrightarrow{u}) \leq f_i(\overrightarrow{v}) \text{ and } \exists i \in \{1, 2, \ldots, k\},\ f_i(\overrightarrow{u}) < f_i(\overrightarrow{v}) \quad (2)$$

In other words, a solution $\overrightarrow{u}$ dominates another solution $\overrightarrow{v}$ (denoted $\overrightarrow{u} \succeq \overrightarrow{v}$) if $\overrightarrow{u}$ is better than $\overrightarrow{v}$ on at least one objective and no worse than $\overrightarrow{v}$ on all the other objectives. The *Pareto optimal set* P_s consists of the set of non dominated solutions: a solution $\overrightarrow{u}$ belongs to P_s if there is no other solution which dominates $\overrightarrow{u}$. A *Pareto optimal front* P_f contains all objective function values corresponding to the solutions in P_s (i.e., each point in P_s maps to one point in P_f). Of course, in this paper we only consider an *approximation* of the Pareto optimal set since P_s is not known. In the following we will not mention any further that our notions of P_s and P_f are approximations of their unknown optimal counterparts.

With respect to our RTT estimation problem, we define two objective functions:

1. We define $ObjectiveFitness_1(\overrightarrow{u})$ as the average of the absolute distances between the sequence of $estimatedRTT$ constructed by solution $\overrightarrow{u}$ and the corresponding sequence of the $measuredRTT$ actually observed.
2. We define $ObjectiveFitness_2(\overrightarrow{u})$ as the number of times in which the $estimatedRTT$ constructed by $\overrightarrow{u}$ is lower than the corresponding $measuredRTT$.

Both functions are evaluated on a set of training data collected as described in section 4.2. The ideal value for each of the two objective functions is zero. In the following subsections we describe the approaches that we have applied to this MOP.

In all the approaches we kept the nondominated solutions found during the evolutionary search. That is, at each generation we perform the following steps: (i) store in an external archive all the individuals nondominated by any other individual in the current population; (ii) drop from the archive individuals dominated by some other member of the archive.

3.2 Scalarization

This approach consists in combining the k objective functions in a single scalar objective F_{ws} to be minimized. The combination consists of a weighed sum of the objective functions with weights fixed a priori [7]:

$$F_{ws} = \sum_{i=1}^{k} w_i \cdot f_i \tag{3}$$

Since the relative importance of the objectives cannot be determined univocally — i.e., with one single choice of weights — we explored several combinations of weights between $[0.0, 2.0]$ varied in steps of 0.25. In the extreme cases we give weight 2 to one of the objectives and weight 0 to the other. Note that the sum of weights is equal to the number of objectives.

3.3 Pareto Dominance

We applied the Pareto dominance technique with a tournament selection scheme close to that in [9]. In the cited paper an individual is randomly picked from the population and then compared with a comparison set. Individuals that dominate the comparison set are selected for the reproduction. We modified this scheme according to the classical tournament selection, in which a group of n ($n \geq 2$) individuals is randomly picked from the population and the one with the best fitness is selected. Our scheme works as follows:

1. A tournament set of n ($n \geq 2$) individuals is randomly chosen from the population.
2. If one individual from the set is not dominated by any other individual, then it is selected.
3. Otherwise an individual is chosen randomly from the tournament set.

4 Experimental Procedure

4.1 RTT Traces

We collected a number of RTT samples on the mail server of our University. This server handles a traffic in the order of 100.000 messages each day (see `http://mail.units.it/mailstats/`). We intercepted the SMTP traffic at the mail server for 10 minutes every 2 hours for 12 consecutive days (SMTP is the application-level protocol for sending email messages). The *tcpdump* software intercepted the network packets. The output of this tool was processed by the *tcptrace* software which constructed the *measuredRTT* data for each connection. We then dropped connections with less than 5 RTT values. The tools that we used are freely available on the web, at `http://www.tcpdump.org/` and `http://www.tcptrace.org/` respectively.

The resulting trace consists of 396109 RTT measures in 41521 TCP connections. These data are grouped in 78 files, for convenience. We chose one of these

files as *training set*, consisting of 5737 RTT measures on 611 TCP connections. The file selected as training set exhibits a large variety of scenarios: small and large variations, abrupt changes and so on. We used the remaining files, consisting of 390372 RTT measures on 40910 TCP connections, as *cross validation set*. The generalization capabilities of the solutions found on the training set have been evaluated on the cross validation set.

4.2 On Setting the GP Process

The terminal set and the function set is shown in Table 1. We used only arithmetic operators and a power of 2 as constant in order to obtain formulas that can be computed efficiently (this is a key requirement in TCP implementations and is necessary for fair comparison with the RTT estimator developed by Jacobson that is currently used). We allow the resulting formula to include the last measured RTT ($measuredRTT_{i-1}$) and the last estimation ($estimatedRTT_{i-1}$). We did not include values more far away in the past because the autocorrelation of RTT traffic is known to decrease very quickly [13].

Table 1. Terminals and functions set

Terminals set	$\frac{1}{2}, 1, measuredRTT_{i-1}, estimatedRTT_{i-1}$
Functions set	$+, -, /, \times$

Table 2. Parameter settings

Parameter	**Setting**
Population size	500
Selection	Tournament of size 7
Initialization method	Ramped Half-and-Half
Initialization depths	2-6 levels
Maximum depth	6
Internal node bias	90% internals, 10% terminals
Elitism	5
Crossover rate	80%
Mutation rate	20%

For the first multi-objective approach we performed 25 independent executions for each combination of weights for a total of 225 runs, for the Pareto tournament scheme we performed the same amount of runs in order to carry out a fair comparison. Each execution starts with a different seed for the random number generator. We allocate 50 generations for each test. All others parameters are summarized in the Table 2. We used Sean Lukes Evolutionary Computation and Genetic Programming Research System (ECJ15) which is freely available on the web at `http://cs.gmu.edu/~eclab/projects/ecj/`. We modified and extended the original API for our needs.

5 Results

5.1 Comparison of the Multi-objective Approaches

In this section we compare the Pareto fronts generated by multi-objective genetic programming search based on the scalarization method and the Pareto-based tournament selection. A thorough comparison between the two approaches would require several indicators as those described in [18,6]. We use a much simpler comparison for lack of space and because both approaches exhibit significantly better performance than our baseline solution — the original algorithm by Jacobson.

We evaluated the performance on the cross validation dataset of the Jacobson algorithm and of each solution found with GP and belonging to the Pareto set. The results are summarized in Figure 4. The most important result is that

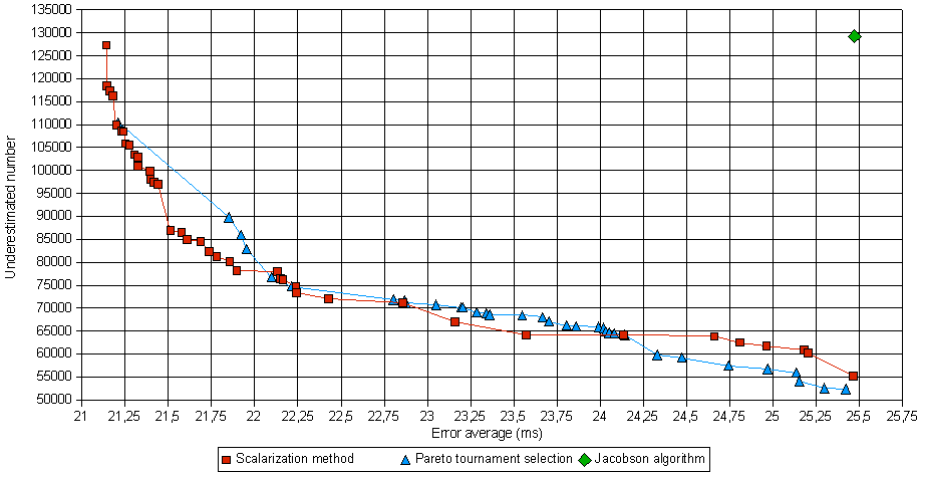

Fig. 4. Pareto front generated with the nondominated solutions for each multi-objective approach

GP found 84 solutions that outperform the Jacobson algorithm, 40 have been found with the scalarization method and 34 with the Pareto based tournament selection. This result is particularly significant because it demonstrates the potential effectiveness of GP in an important application domain. Interestingly, the scalarization method generates solutions that dominate those found with the Pareto-based tournament for a large range of values of the error average (from 21.125 to 24.125) except for one solution. Pareto based tournament provide better solutions in terms of the number of underestimated RTTs.

5.2 Comparison of the RTT Estimators

To gain further insights into the quality of the solutions, in particular regarding the improvements that can be obtained with respect to the Jacobson algorithm,

we analyzed the performance on each single file of the cross validation dataset. We present the results for the Jacobson algorithm and for two solutions located at the two extremes of the Pareto front: the one giving the best results in terms of underestimated RTTs (located bottom-right in Figure 4) and the one giving the best results in terms of average error (located top-left).

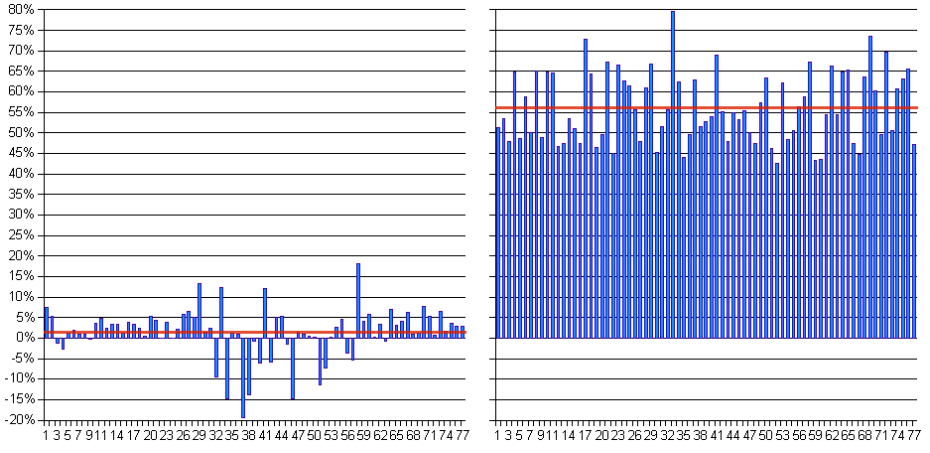

Fig. 5. Number of underestimated RTT for each trace file. Best formula found by GP in terms of average error (left) and in terms of number of underestimated RTTs (right)

Figure 5 describes the results in terms of underestimated RTTs. Each point in the X-axis corresponds to one file of the cross validation dataset, whereas the Y-axis is the improvement with respect to Jacobson, in percentage. The horizontal line represents the average improvement across the entire cross validation dataset. Figure 5-left shows the result for the best formula found by GP in terms of average error. It can be seen that the average improvement over the Jacobson algorithm is small (approximately 2%) and that in some files the average error is worse. Figure 5-right shows the result for the best formula found by GP in terms of number of underestimates. This case is much more interesting because the formula found by GP largely outperforms the Jacobson algorithm, with a 56% average improvement (34% of RTTs are underestimated by Jacobson and only 15% by the formula found by GP). Moreover, a remarkable improvement can be observed in every trace file.

Figure 6 describes the results in terms of average error, with the same notation as above. It can be seen the best formula in terms of average error (left figure) exhibit a 15% average improvement over Jacobson (corresponding to 4.7 ms) and that some improvement can be observed in every trace file. The best formula in terms of underestimated RTTs (right figure) exhibits instead essentially the same performance as that of Jacobson.

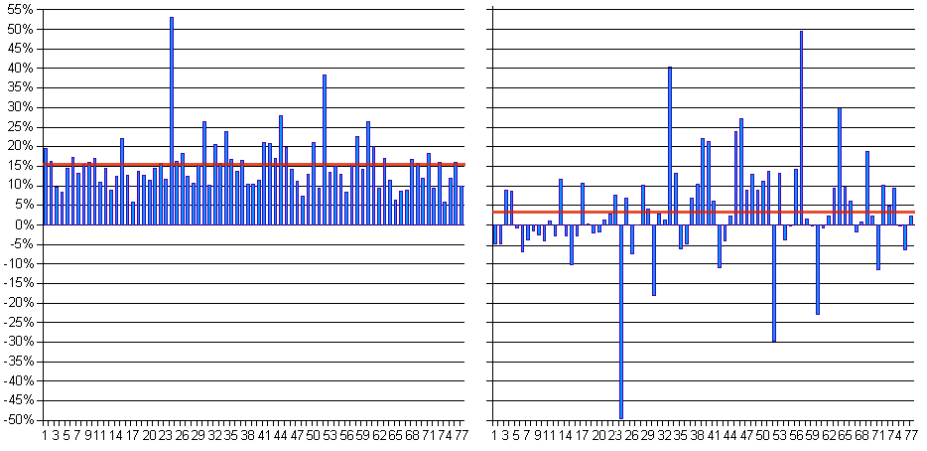

Fig. 6. Error average for each trace file. Best formula found by GP in terms of average error (left) and in terms of number of underestimated RTTs (right)

6 Concluding Remarks

We applied two radically different multi-objective approaches on an important real-world problem. We used an *a priori* method which combines all the objectives into a single one by weighting each objective in advance, and an *a posteriori* approach based on a Pareto tournament selection. The quality of the solutions provided by the two approaches is similar, although solutions obtained with Pareto-based tournament tended to be more effective in terms of the number of underestimated RTTs (a particularly critical issue for TCP performance). The effectiveness of the simple scalarization method was rather surprising and is probably due to the small number of objectives: covering a sufficiently wide set of weights remains computationally acceptable.

While this is an interesting result itself, the most significant result consists in the performance of the formulas found with multi-objective GP: they are significantly better than those exhibited by the RTT estimator used in all TCP implementations. This result could lead to several interesting applications of GP in the networking field — e.g., tailoring RTT estimators to individual hosts, rather using the same estimator for all hosts; differentiating the estimator based on the application using TCP, whether web navigation or transmission of email; and so on.

Acknowledgments

The authors are grateful to Stefano Catani from the Centro Servizi Informatici di Ateneo (CSIA) for his technical help. This work is supported by the Marie-Curie RTD network AI4IA, EU contract MEST-CT-2004-514510 (December 14th 2004).

References

1. RFC 793, "Transmission Control Protocol,", Sept. 1981.
2. RFC 1122, "Requirements for Internet Hosts - Communication Layers," Oct. 1989.
3. J. Aikat, J. Kaur, F. D. Smith, K. Jeffay, "Variability in TCP round-trip times," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 279–284, 2003.
4. M. Allman, V. Paxson, "On estimating end-to-end network path properties," in *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 29, pp. 263-274, Oct. 1999.
5. W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, "Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications," Morgan Kaufmann, dpunkt.verlag, 1998.
6. P. A. N. Bosman, D. Thierens, The Balance Between Proximity and Diversity in Multiobjective Evolutionary Algorithms, in *Evolutionary Computation, IEEE Transactions*, Vol. 7, no. 2, pp. 174-88, Apr. 2003.
7. C. A. Coello Coello, "An Updated Survey of GA-based Multiobjective Optimization Techniques," in *ACM Computing Surveys*, Vol. 32, No. 2, Jun. 2000.
8. C. A. Coello Coello, D. A. Van Veldhuizen, G. B. Lamont, "Evolutionary Algorithms for Solving Multi-Objective Problems," Kluwer Academic Press, 2002.
9. A. Ekárt, S.N. Németh, "Selection Based on the Pareto Nondomination Criterion for controling Code Growth in Genetic Programming," in *Genetic Programming and Evolvable Machine*, Vol. 2, March 2001, pp. 61–73.
10. C. M. Fonseca, P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proceedings 5th Int. Conf. Genetic Algorithms*, Ed. San Mateo, Morgan Kaufmann, pp. 416-423, 1993.
11. V. Jacobson, "Congestion avoidance and control," in *Proceedings ACM SIGCOMM*, Stanford, pp. 314-329, Aug. 1988.
12. J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection," *MIT Press*, 1992.
13. L. Ma, K. E. Barner, G. R. Arce, "Statistical Analysis of TCP's Retransmission Timeout Algorithm," in *IEEE/ACM Transactions on Networking*, Vol. 14, Issue 2, Apr. 2006.
14. D. Parrott, X. Li, V. Ciesielski, "Multi-objective Techniques in Genetic Programming for Evolving Classifiers," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 1141-1148, IEEE Press, 2-5 September 2005.
15. C. Poloni, "Hybrid Genetic Algorithm for Multiobjective Aerodynamic Optimisation," in *Genetic algorithms in engineering and computer science*, John Wiley & Sons, pp. 397–415, 1995.
16. N. Srinivas, K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," in *Evolutionary Computation*, Vol. 2, no. 3, pp. 221-248, 1994.
17. W. R. Stevens, G. R. Wright, "TCP/IP illustrated (vol. 2): the implementation", Addison-Wesley Longman Publishing Co., Inc., Boston, 1995.
18. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. Fonseca, Performance Assessment of Multiobjective Optimizers: An Analysis and Review, in *Evolutionary Computation, IEEE Transactions*, Vol. 7, no. 2, pp. 117-132, Apr. 2003.