An Adaptive Random Search Alogrithm for Optimizing Network Protocol Parameters

Tao Ye, Shivkumar Kalyanaraman
Department of Electrical, Computer and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180
yet3@networks.ecse.rpi.edu, shivkuma@ecse.rpi.edu

Abstract

The optimization of network protocol parameters based on network simulation can be considered a "black-box" optimization problem with unknown, multi-modal and noisy objective functions. In this paper, an adaptive random search algorithm is proposed to perform efficient and robust optimization for the concerned problems. Specifically, the algorithm is designed for use by the on-line simulation scheme which attempts to automate network management with protocol parameter tuning. The new algorithm takes advantage of the favorable statistical properties of pure random search and achieves high efficiency without imposing extra restriction, e.g., differentiability, on the objective function. It is also robust to noises in the evaluation of objective function since no traditional local search technique is involved. The proposed algorithm is tested on some classical benchmark functions and its performance compared with some other stochastic algorithms. Finally, a real case test is presented, in which the parameters of some RED queues are optimized with our algorithm.

1 Introduction

Optimization problems arise in many engineering areas and can be formulated as (assume minimization): given a real-valued objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, find a global minimum,

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in D} f(\mathbf{x}) \tag{1}$$

where D is a compact set in $\mathbb{R}^n$ and called parameter space. In these problems, the objective functions $f(\mathbf{x})$ are usually analytically unknown and the function evaluation can only be achieved through computer simulation or other indirect ways. Hence, these problems are called "black-box" optimization. Since their objective functions are often non-linear and multi-modal, they are also termed global optimization in contrast with local optimization where there is only one single extreme in $f(\mathbf{x})$. These optimization problems are considered very hard to solve.

An on-line simulation scheme has been proposed in [1], which attempts to accomplish automatic and adaptive network management with an approach based on "black-box" parameter optimization. The basic idea of this scheme is illustrated in Fig 1. In this scheme, autonomous on-line simulators in each network domain continuously collect real-time data on network conditions, run network simulations and use some optimization algorithm to come up with better parameters for network protocols or algorithms, such as RED, OSPF. A good optimization algorithm, which can quickly find appropriate protocol parameters, is essential to the success of this scheme. Besides the difficulties common in all "black-box" optimization problems, some specific properties in network optimization need to be considered in designing a suitable algorithm.

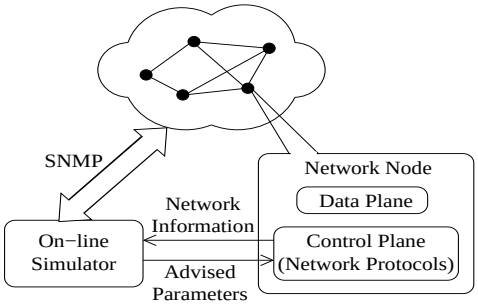

Figure 1: On-line simulation scheme for automatic network management

First, since network conditions keep changing all the time, a very efficient algorithm is required, which is able to quickly find better network parameters before significant changes happen to network conditions on which network simulation is based. Accordingly, the emphasis of the algorithm should not be on seeking the optimum setting, but finding a better operating point within the limited time frame. The high efficiency requirement is also due to the fact that the simulation of a complex network is often very time-consuming and it is necessary to reduce the number of function evaluations as much as possible. Secondly, since a network often has tens or hundreds of protocol parameters to be tuned, the desired algorithm should also be capable of handling a very large parameter space. Thirdly, network simulation only provides us with approximate estimation of network performance, which means the objective functions in our problems are superimposed with small random noises due to inaccuracy in network modeling, simulation, etc. In addition, since a network normally has both discrete and continuous parameters, we should also design the search algorithm to handle this situation.

In recent years, much effort has been contributed to the area of "black-box" optimization, and many algorithms have been proposed. Among them, stochastic algorithms have achieved wide popularity with successful application in many areas. These algorithms introduce stochastic elements into the search process and hence can only guarantee a probabilistic convergence to the global optimum. Their success is mostly due to the fact that they assume little knowledge on the objective functions and are often simple and easy to implement. As a result, they can be adapted to handle large-scale and complex problems without much difficulty. Pure random search is the simplest stochastic algorithm, which performs uniform sampling in the parameter space. Although it is not widely used for its obviously poor efficiency, random search has become the basis of many algorithms to guarantee the probabilistic convergence to the global optimum. Controlled random search[2] and its variants[3, 4, 5] replace random sample with simplex sampling technique. The basic idea is to first uniformly sample the parameter space and then increasingly bias the sampling towards the good regions found in the previous search. Multistart hillclimbing tries to improve the efficiency of random search by starting a hillclimbing search from each random sample until reaching a local optimum. However, it may waste a lot of time on re-climbing the hills already visited. Clustering method[6] attempts to avoid the revisit of the hills by identifying the random samples close to each other as a cluster and only starting one hillclimbing process from each cluster in the hope that a cluster only includes one local optimum. However, accurately identifying these clusters is difficult, especially in high dimension problems[6]. Simulated annealing[7] and Genetic algorithm[8] are two widely used techniques inspired by natural phenomenon, i.e., physical annealing process and natural evolution process. Although they have been demonstrated to be effective in many practical problems, their efficiencies are usually low and the algorithms often take a long time to converge. Many variants have been tried to combine them with some local search methods to improve their efficiencies.[9]

No Free Lunch theorem[10] has demonstrated that no single algorithm can consistently perform better

in all classes of problems than the other algorithms. That is, for one class of problems where an algorithm can do better than other algorithms, there is always another class where it will do worse. For one specific optimization problem, the most efficient algorithms are those which best exploit the available information of the objective function. This paper examines the properties of network optimization problems and proposes an adaptive random search algorithm which is geared to the requirements and restrictions of network optimization problems and attempts to achieve high efficiency with little *a priori* knowledge assumed.

The rest of this paper is organized as follows: in Section 2, we discuss design ideas of the search algorithm in the context of network optimization and investigate the applicability of various search techniques. In Section 3, we present the new algorithm which is designed on the basis of discussions in previous sections. In Section 4, the proposed algorithm is tested on some benchmark functions, and its performance compared with some other stochastic algorithms. We also apply the algorithm to the real network optimization experiment and show how the performance of network can tuned by our search algorithm. Finally, we conclude this paper in Section 5 and discuss further research directions.

2 Design Ideas of Search Algorithm for Network Optimization

A stochastic search algorithm consists of two procedures: *exploration* and *exploitation*, which are also referred to as *global phase* and *local phase* in some literature[11, 6]. Exploration encourages the search process to examine unknown regions, while exploitation attempts to converge to a local optimum in the vicinity of a chosen region. They can be executed alternately or be interwoven together during the search. The balance between these two procedures is maintained by a *balance strategy* which can be either probabilistic or deterministic. For example, multistart hillclimbing uses random sampling for exploration, a local search technique for exploitation and a deterministic balance strategy to execute the two procedures alternately. Since no existent algorithm can be directly used to achieve high efficiency without exploiting any problem-specific information, our approach is to investigate the applicability of various search techniques, choose those which best fit our problem, and then use an appropriate balance strategy to combine these methods and achieve high efficiency.

Exploration methods are usually required to provide a guarantee of a probabilistic convergence to the global optimum. Not many techniques are available in this category. Some of the examples are deterministic enumeration, random sampling, random walk, etc. Among them, random sampling may be the simplest and most widely used technique, which takes random samples from a uniform distribution over the parameter space. This technique provides a strong probabilistic convergence guarantee and is more efficient than deterministic methods, especially for high-dimension problems[6]. Therefore, we have adopted random sampling for exploration in our algorithm. Although it is generally considered to lack in efficiency, random sampling is in fact very efficient in identifying a good point close to promising areas in its initial steps. To illustrate this, we first define a measure:

$$\phi_D(y_r) = \frac{m(\{\, \mathbf{x} \in D \mid f(\mathbf{x}) \leq y_r \,\})}{m(D)} \tag{2}$$

where $m(\cdot)$ is *Lebesgue measure*. Then we can define a *r-percentile* region in the parameter space D:

$$A_D(r) = \{\, \mathbf{x} \in D \mid f(\mathbf{x}) \leq y_r \,\} \tag{3}$$

where

$$\phi_D(y_r) = r, \quad r \in [0, 1]$$

y_r is called *r-percentile* of the objective function value. Note that $A_D(1)$ is just the whole parameter space D and $\lim_{\epsilon \to 0} A_D(\epsilon)$ will converge to the global optimum. Suppose the sample sequence generated by n

steps of random sampling is $\mathbf{x}_i, i = 1 \ldots n$ and $\mathbf{x}^n_{(1)} = \arg\min_{1 \le i \le n} f(\mathbf{x}_i)$, then the probability of $\mathbf{x}^n_{(1)}$ in $A_D(r)$ is:

$$P(\mathbf{x}^n_{(1)} \in A_D(r)) = 1 - (1 - r)^n \tag{4}$$

We can have the r value of the *r-percentile* region that $\mathbf{x}^n_{(1)}$ will reach with probability p is:

$$r = 1 - (1 - p)^{1/n} \tag{5}$$

For any probability $p < 1$, r will tend to 0 with increasing n. This is just the global convergence guarantee of random sampling. From Equation 5, we can also see that random sampling is highly efficient at initial steps since r decreases exponentially with increasing n, and its inefficiency is from later samples. For example, Fig 2 shows the convergence curve of random sampling with a probability of 0.99. We can see that it takes only 44 samples to reach a point in $A_D(0.1)$ area and this is achieved without using any special information of the objective function, whereas all samples after the first 44 can only improve r value of $\mathbf{x}^n_{(1)}$ at most by 0.1.

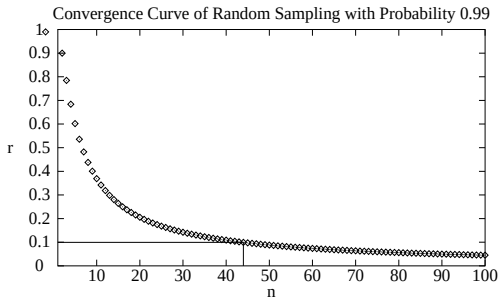

Figure 2: Convergence curve of random sampling with probability 0.99

Exploitation methods are required to quickly converge to the local optimum in a certain region. The derivative-based methods, such as quasi-Newton method and deepest descent, are not adopted in our algorithm though they are very efficient for differentiable objective functions. The reason is that the objective function in our problem may be superimposed with noises and hence the accurate derivative estimation cannot be obtained. Direct search methods, which do not use any derivative information of objective functions, are widely used in practice and often recommended for optimization in the presence of noise[5, 12]. However, like other local search algorithms, the performance of traditional direct search methods, such as Nelder-Mead simplex method[13] and pattern search[14], are still susceptive to the effect of noises and may be degraded in higher dimension problems[15, 6]. Therefore, we have used a new direct search approach in our algorithm which is based on the high efficiency of random sampling at initial steps. This approach starts a random sampling in the prospective region with a relatively large sample space, then shrinks and re-aligns the sample space based on previous samples until it finally converges to the local optimum. The details of this approach are described in the next section. Since only random sampling is used, our method is more robust to noises in function evaluations and more scalable. These advantages are achieved without sacrificing the efficiency since the method takes advantage of the initial high-efficiency property of random sampling.

The balance strategy is essential to the efficiency of an algorithm. Usually it should only execute the exploitation procedure in prospective areas. However, it is difficult to judge which areas are more promising and should be exploited. Some algorithms, such as multistart type algorithms, do not differentiate areas and hence lack in efficiency. Our approach is to identify a certain *r-percentile* region $A_D(\gamma)$ and only start exploitation from the points in this region. The size of $A_D(\gamma)$ region should be small enough so that most of

unpromising areas are filtered out, and on the other hand, it should be within the reach of initial high efficient steps of random sampling so that identifying a point in it will not take too long and thus lower the overall efficiency. Fig 3 illustrates an example of this strategy. The left plot shows a contour plot of a 2-dimension multi-modal objective function and the right plot shows the region of $A_D(0.05)$. We can see the function has many local optima; however, only 3 discrete areas remain in $A_D(0.05)$ (shaded areas in the right plot). Each of these areas encloses a local optimum and the one with the biggest size has the global optimum. This is also true for most optimization problems since the region of attraction for the global optimum is usually the largest[11]. If we do random sampling on the whole parameter space, those samples falling in $A_D(\gamma)$ are also uniformly distributed on $A_D(\gamma)$. As a result, if we start exploitation from these points, the search will arrive at a global optimum with a larger probability than other non-global optima.

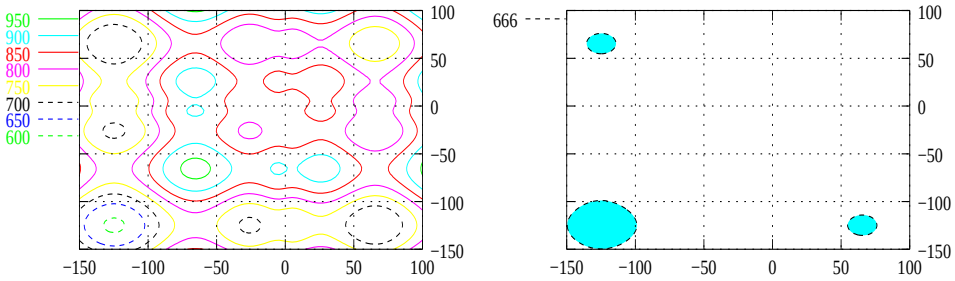

Figure 3: Contour plot of an objective function(left) and its region of $A_D(0.05)$(right)

As we mentioned before, our algorithm needs to be able to handle the objective functions with a mixture of continuous and discrete parameters. Since most discrete parameters in network protocols, such as queue size limit, have a natural continuous analog, we have used two methods introduced in [16] for these parameters. The first method expands a discrete parameter space to a continuous one by making the function value of a non-discrete point equal to that of its nearest discrete neighbor. The second method modifies the search algorithm so that the algorithm can only generate discrete points, and one way to do this is rounding each non-discrete point generated by the algorithm to its nearest discrete neighbor

3 The Algorithm

Based on the analysis in the previous section, we have designed an adaptive random search algorithm for network optimization problems. The basic idea of the algorithm is: use random sampling to explore the whole parameter space and only start exploitation for those points which fall in a certain $A_D(\gamma)$ region. To identify an $A_D(\gamma)$ area, we should first decide the threshold function value y_γ, and any point with a smaller function value than y_γ belongs to $A_D(\gamma)$. Our algorithm uses the following strategy to decide the value of y_γ: first we choose a relatively small value $r \in [0, 1]$, and a confidence probability p, then we take n random samples where n is the smallest integer satisfying $n \geq \frac{\ln(1-p)}{\ln(1-r)}$ so that the probability of $\mathbf{x}^n_{(1)} \in A_D(r)$ is larger than p. We use $f(\mathbf{x}^n_{(1)})$ as the initial value of y_γ. In the future exploration, we obtain a new $\mathbf{x}^n_{(1)}$ every n samples and update y_γ with the average function value of these points. In this method, the confidence probability p should assume a value close to 1, for example, 0.99. The value of r decides the value of γ (γ is smaller than r) and hence the size of $A_D(\gamma)$. It should be chosen carefully to balance efficiency and effectivity as discussed before. We have used $r = 0.1$ in our current algorithm, and in this case it only takes 44 samples to find a point which falls within $A_D(0.1)$ with a probability larger than 0.99.

Now we will describe our exploitation technique which is also based on random sampling. We start an exploitation process for the points with a function value smaller than y_γ. According to previous discussions, the starting point $\mathbf{x}_0$ is close to a certain local optimum. Suppose we have a neighborhood $N(\mathbf{x}_0)$ of this point which also includes the local optimum, if $\phi_{N(\mathbf{x}_0)}(f(\mathbf{x}_0))$ is large, many points in $N(\mathbf{x}_0)$ are better than $\mathbf{x}_0$. Therefore, if we do random sampling in $N(\mathbf{x}_0)$, it will very probably find a point better than $\mathbf{x}_0$ with a small number of samples. Assuming $\phi_{N(\mathbf{x}_0)}(f(\mathbf{x}_0)) = \upsilon$, with a confidence probability q, random sampling should find a better point in $N(\mathbf{x}_0)$ with $l = \frac{\ln(1-q)}{\ln(1-\upsilon)}$ samples. If a better point is found in l samples, we replace $\mathbf{x}_0$ with this point, move the sample space to the new $N(\mathbf{x}_0)$ and keep its size unchanged. In this way, even if $N(\mathbf{x}_0)$ may miss the local optimum, the consecutive descent moves will still lead the search to converge to the local optimum in a way similar to the local search. If the above method fails to find a better point in l samples, that suggests $\phi_{N(\mathbf{x}_0)}(f(\mathbf{x}_0))$ is small. In this case, we contract $N(\mathbf{x}_0)$ by a certain ratio $c \in [0, 1]$, i.e., generate a new neighborhood $N'(\mathbf{x}_0)$ whose size is $c \cdot m(N(\mathbf{x}_0))$. With this method, the search will soon converge to the local optimum, or to an appropriate neighborhood where $\phi_{N(\mathbf{x}_0)}(f(\mathbf{x}_0))$ is υ, then moves to the local optimum. The exploitation process continues until $m(N(\mathbf{x}_0))$ is less than a certain threshold, whose value is dependent on the resolution requirement of the optimization problem. In our current implementation, the initial size of $N(\mathbf{x}_0)$ is taken as $r \cdot m(D)$, where D is the original parameter space and r the parameter in exploration process. Since our exploration guarantees that with a high probability $\mathbf{x}_0$ belongs to a certain $A(\gamma)$ with a size smaller than $r \cdot m(D)$, the above initial size will ensure $N(\mathbf{x}_0)$ covers a local optimum in most cases. A simple method is currently used to construct $N(\mathbf{x}_0)$: assuming a n-dimension parameter space S defined with $l_i \leq x_i \leq u_i, i = 1 \ldots n$, the neighborhood of $\mathbf{x}_0$ with a size of $r \cdot m(S)$ is: $N_{S,r}(\mathbf{x}) = \{z \in S \mid |z_i - x_i| < r^{1/n} \cdot (u_i - l_i)\}$.

The detail of the algorithm is illustrated by the following pseudo-codes:

Algorithm 1 Adaptive Random Search Algorithm

Initialize exploration parameters p, r, $n \leftarrow \ln(1-p)/\ln(1-r)$
Initialize exploitation parameters q, υ, c, s_t, $l \leftarrow \ln(1-q)/\ln(1-\upsilon)$
Take n random samples from parameter space D
$\mathbf{x}_0 \leftarrow \arg\min_{1\le i\le n}(f(\mathbf{x}_i))$, $f_\gamma \leftarrow f(\mathbf{x}_0)$, add $f(\mathbf{x}_0)$ to $\mathbf{F}$,
$i \leftarrow 0$, $flag \leftarrow 1$, $\mathbf{x}_{opt} \leftarrow \mathbf{x}_0$
while stopping criterion is not satisfied
  if $flag = 1$
    $j \leftarrow 0$, $f_l \leftarrow f(\mathbf{x}_0)$, $\mathbf{x}_l \leftarrow \mathbf{x}_0$, $\rho \leftarrow r$
    while $\rho > s_t$
      Take a random sample $\mathbf{x}'$ from $N_{S,\rho}(x_l)$,
      if $f(\mathbf{x}') < f_l$
        $\mathbf{x}_l \leftarrow \mathbf{x}'$, $f_l \leftarrow f(\mathbf{x}')$,
        $j \leftarrow 0$
      else
        $j \leftarrow j + 1$
      end
      if $j = l$
        $\rho \leftarrow c \cdot \rho$, $j \leftarrow 0$
      end
    end
    $flag \leftarrow 0$, update $\mathbf{x}_{opt}$ if $f(\mathbf{x}_l) < f(\mathbf{x}_{opt})$
  end
  Take a random sample $\mathbf{x}_0$ from S,
  if $f(\mathbf{x}_0) < f_\gamma$
    $flag \leftarrow 1$
  end
  if $i = n$
    Add $\min_{1\le i\le n}(f(\mathbf{x}_i))$ to $\mathbf{F}$
    $f_\gamma \leftarrow$ mean($\mathbf{F}$), $i \leftarrow 0$
  end
  $i \leftarrow i + 1$
end

4 Test Results

We test the performance of our search algorithm on two benchmark functions, which are shown in Fig 4. The first one is a modified version of 2-dimension Rastrigin function defined on $x, y \in [-10, 10]$, which has a global minimum at $[2.5, 2.5]$ with a function value of 0. This function has a large number of local optima and is considered very difficult to optimize. From its surface plot, we can see this function is similar to a sphere with small noises superimposed. The second one is Schwefel function defined on $x, y \in [-150, 100]$ and has a global minimum at $[-125, -125]$ with a value of 592. This function also has quite a few local minima, especially two near global optima far from the real one.

We apply our search algorithm to these two functions, repeat each test with randomly selected starting points for 50 times and take the average of the results from these tests. Based on these results, we plot the convergence curve to study the convergence speed of the algorithm. We also apply two other search

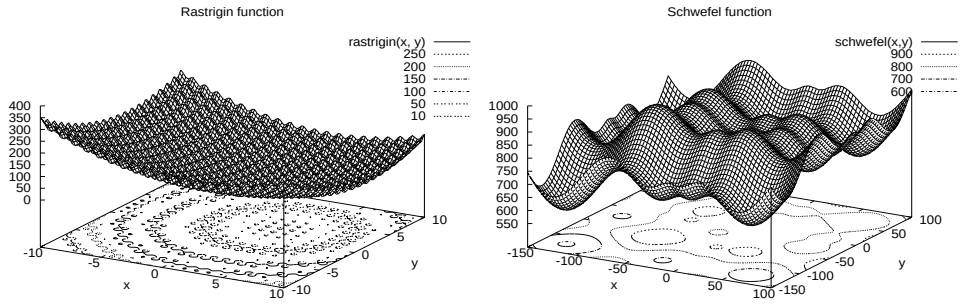

Figure 4: Rastrigin and Schwefel functions

algorithms to the test functions and compare their performance with that of our algorithm. One is a multistart hillclimbing algorithm using a version of pattern search as its exploitation technique. The other is a version of improved multistart hillclimbing[17] which starts the exploitation process only after finding a point better than the best one found so far and thus avoids the revisit problem of regular multistart algorithms. These two algorithms are easy to implement and widely used, and they are also more scalable to high-dimension problems than those more sophisticated algorithms such as clustering method. In the tests, we have used in our algorithm the following parameters:$p = 0.99, r = 0.1, c = 0.5, q = 0.99, s_t = 0.01$. The test results are displayed in Fig 5. We can see that our algorithm converges to the global optimum rapidly and

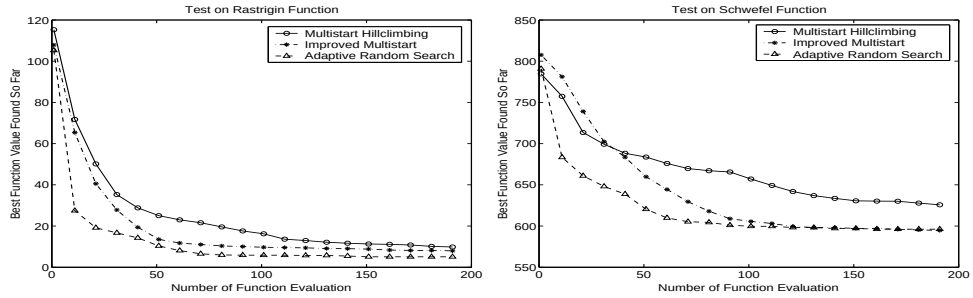

Figure 5: Test results of search algorithms

its efficiency is much better than two other algorithms under test. For Rastrigin function where the global optimum is hidden in a large number of neighboring local optima (noises), our algorithm can still find the points very close to the global optimum which the other two algorithms can not reach. This demonstrates that our algorithm is robust to the effect of noises.

We also test our algorithm on real network optimization problems. We build a Linux-based testbed with the topology shown in Fig 6. There are 4 routers with a Random Early Drop (RED) queue on each of them and each RED queue has 4 parameter to tune: *min threshold*, *max threshold*, *max drop probability* and *exponential weighted moving average coefficient*. Thus we have a total of 16 parameters to tune in this experiment. We generate some TCP flows from one side to the other and use as the performance metric the coefficient of variation (standard deviation over mean) of goodputs for these TCP flows, which measures the variability of the goodputs. An on-line simulator is applied in the network to tune the parameters of these RED queues. This on-line simulator uses *ns*[18] to simulate this network and our algorithm to search for better RED parameter settings. If a parameter setting is found with a performance metric better than the current one by a certain margin, it is applied back into the network. The COV of goodputs during the experiment is plotted in Fig 7. In the beginning, the parameters of these RED queue are purposedly set to

some bad values, which would result in a great unfairness of goodputs between TCP flows. We can see a high average COV value and big oscillations. Then the on-line simulator is started. Appropriate parameter settings are quickly found by our search algorithm and the previously mis-configured parameters corrected, which results in an immediate performance improvement: the average COV drops to a very low value and the COV curve becomes very stable over time.

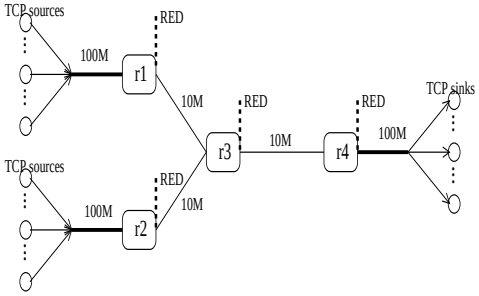

Figure 6: A 4-router testbed configuration

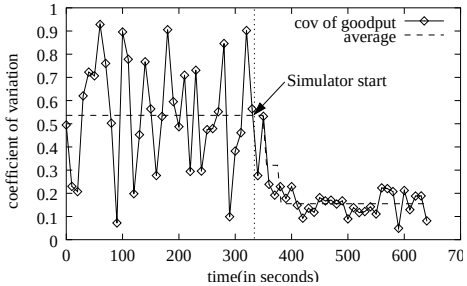

Figure 7: Optimize coefficient of variation of goodputs for TCP flows

5 Conclusion

In this paper, we investigate the optimization problem of network protocol parameters, which can be considered a simulation-based "black-box" optimization problem with noisy, multi-modal objective functions. The search algorithm is required to be very efficient, robust to noises and scalable to high dimension problems. Classical optimization algorithms, such as controlled random sampling, clustering method and simulated annealing, do not fit well in our problem with all these restrictions. An adaptive random search algorithm is proposed, which is mainly based on random sampling and hence robust to noises and scalable to high-dimension problems. High efficiency is also achieved by exploiting the advantageous statistical property of random sampling. The algorithm is tested on some benchmark functions and real network optimization problems and the test results have indicated that our algorithm performs very efficiently and is robust to noises.

Currently the tests are performed on relatively small-scale problems. Future work should study the scalability of our algorithm and test it on large-scale problems, such as BGP routing algorithm which may have hundreds of parameters. Besides, the algorithm can be further enhanced to achieve higher efficiency. For example, tabu technique can be included to reduce the probability of revisits. The parallelization of the algorithm is another research direction to handle large-scale optimization problems.

References

[1] Tao Ye, David Harrison, and et. al. Traffic management and network control using collaborative on-line simulation. In *Proc. of IEEE ICC'01*, Helsinki, Finland, June 2001. To appear.

[2] W. L. Price. A controlled random search procedure for global optimization. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimization 2*, pages 71–84. North-Holland, Amsterdam, Holland, 1978.

[3] W. L. Price. Global optimization by controlled random search. *Journal of Optimization Theory and Applications*, 40:333–348, 1978.

[4] Montaz Ali, Aimo Törn, and Sami Viitanen. A numerical comparison of some modified controlled random search algorithm. Technical report, Turku Centre for Computer Science, Turku, Finland, March 1997.

[5] P. Brachetti, M. De Felice Ciccoli, G. Di Pillo, and S. Lucidi. A new version of the price's algorithm for global optimization. *Journal of Global Optimization*, 10:165–184, 1997.

[6] A. H. Kan and G. T. Timmer. Stochastic global optimization methods part I: Clustering methods. *Mathematical Programming*, 39:27–78, 1987.

[7] S. Kirkpatrick, D.C. Gelatt, and M.P. Vechhi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[8] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

[9] W. E. Hart. *Adaptive Global Optimization with Local Search*. PhD thesis, University of California, San Diego, 1994.

[10] D. h. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction on Evolutionary Computing*, 1:67–82, 1997.

[11] Aimo Törn and Antanas Žilinskas. *Global Optimization*. Lecture Notes in Computer Science. Springer-Verlag, 1989.

[12] Michael W. Trosset. On the use of direct search methods for stochastic optimization. Technical report, Department of Computational and Applied Mathematics, Rice University, 2000.

[13] R. MEAD and J. A. NELDER. A simplex method for function minimization. *Comput. J.*, 7(4):308–313, 1965.

[14] R. Hooke and T. Jeeves. Direct search solution of numerical and statistical problems. *J.Assoc. Comp. Mach.*, 8(2):212–229, April 1961.

[15] Soraya Rana, L. Darrell Whitley, and Ronald Cogswell. Searching in the presence of noise. In H. Voigt, W. Ebeling, I. Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV (Berlin, 1996) (Lecture Notes in Computer Science 1141)*, pages 198–207, Berlin, 1996. Springer.

[16] Zelda B. Zabinsky. Stochastic methods for practical global optimization. *Journal of Global Optimization*, 13:433–444, 1998.

[17] J. K. Hartman. Some experiments in global optimization. *Naval Research Logistic Quarterly*, pages 569–576, 1973.

[18] NS. network simulator. http://www-mash.cs.berkeley.edu/ns.