This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Network Configuration and Management via Two-Phase Online Optimization

Bilal Gonen
University of Nevada, Reno
Email: gonenb@cse.unr.edu

Murat Yuksel
University of Nevada, Reno
Email: yuksem@cse.unr.edu

***Abstract*—Automated configuration and management of highly dynamic networks is a challenging problem for network practitioners. Such online optimization of systems can be performed in two ways: (i) using a separate model of the system for experimenting new configurations, (ii) using the system itself for experimentation without a separate system model. The former approach fails for dynamic networks with high failure rates or variable demand profile. In this paper, we take the latter approach and perform in-situ trials in a network to find better configurations of IGP link weights giving result to higher network throughput. Our approach follows a two-phase model where the online optimization process periodically goes into a "search" phase followed by network operation with the parameters found in the latest search phase. We use a black-box optimization algorithm, called Probabilistic Trans-Algorithmic Search (PTAS), to search for better IGP link weights and evaluate our approach in terms of key parameters such as search phase frequency and length.**

I. Introduction

As the size, dynamism and diversity of network components are increasing, the problem of finding the best configuration settings for networks is becoming more difficult. Manageability of large-scale networks is highly dependent on tools and methods to configure them. The typical practice has been to train and employ highly-experienced human administrators for network configuration and management. However, automated ways of managing and configuring networks are essential for scaling this practice for highly dynamic, heterogenous and large networks.

Thus, tools and methods to achieve automated management and configuration of a running network are vitally needed. Being able to tune and configure a running network or system requires one to perform "online" optimization by searching and finding better parameter configuration settings of the system at hand. Such online optimization of systems can be performed in two ways: (i) *using a separate model of the system* for experimenting and trying new configurations, or (ii) *using the system itself* for experimentation without a separate system model. The former approach is appropriate for the cases when the system is stable and exhibits the same or similar behavior over long periods of time. Such stable systems can be characterized by accurate models. For instance, backbone networks or wireless mesh networks with stable links and non-variant traffic profile give pretty fixed response, and can, thus, be accurately modeled over long time periods. Prior research explored online optimization of such networks with for very hard configuration problems such as Interior Gateway Protocol (IGP) link weight setting for load balancing [1], Border Gateway Protocol (BGP) configuration for outbound traffic engineering, and queue parameter setting for network delay minimization.

However, for systems with highly dynamic behavior, it is not practical to accurately capture system's response with a model. Modeling of the real system becomes impractical since the system behavior changes frequently. Thus, the latter approach is needed for such systems. For large networks with high link/router failure rate or a very dynamic traffic demand profile, or mobile ad-hoc networks (MANETs) with a constantly changing topology, it is a necessity to use the system itself to try out new configurations with the hope of finding a better parameter setting. A key challenge in this particular case is the fact that the objective function changes during the optimization itself. Thus, the optimization process should be able to adapt to the changes in the objective function by applying the right search algorithms for newly arising system response due to the system dynamics.

In this paper, we focus on the latter approach and investigate a two-phase online optimization framework for network configuration and management. Our approach follows a pattern of two subsequent phases, *search* and *no-search*, where new configuration parameters are tried during the search phase. Each search phase is followed by a no-search phase with normal operation using the parameters found in the latest search phase. We explore some of the key tradeoffs in the two-phase model, such as how frequent the search should be done, how long should the search phase be, and how worse the search phase can temporarily make the system performance due to its trials. We leverage our prior work, PTAS [2], on hybrid black-box optimization to adapt the search for changing network behavior. We apply the two-phase optimization model on the IGP link weight setting problem and observe the aggregate network throughput against several parameters such as link failure rate, optimization algorithm, and total experiment budget the search phases have.

The rest of the paper is organized as follows: In Section II, we cover prior work related to online optimization of a running system and the IGP link weight setting problem. In Section IV, we outline our two-phase online optimization framework. Section V presents application of our framework on the IGP link weights setting problem. We, then, summarize our work

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

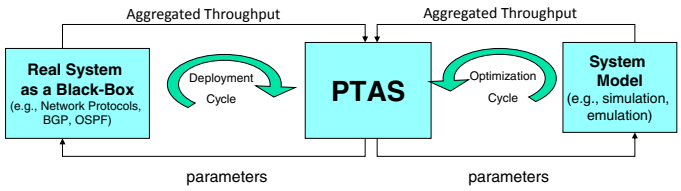


Fig. 1. Optimization using a separate model of the system

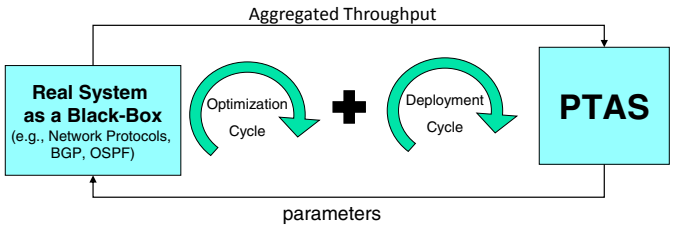


Fig. 2. Optimization using real-time running system

in Section VI.

II. Related Work

System optimization is a vast field with a significant body of prior work. There have been many studies exploring methods for system optimization, of which many have been applied to real-world problems. However, not all of these techniques are suitable for online optimization. The most common design approach of these techniques is to be *greedy and exploitive*, e.g., steepest descent or Newton's gradient methods [3]).

Most real-world applications of online optimization have dealt with software or web systems with a *passive and sample-based* approach [4] to model the recent behavior of the system at hand. Boutilier et al. [5] described a general model of expressive ad contracts of a clearing system and a periodically run channel allocation optimization in real time via online samples. Diao et al. [6] introduced an architecture and its algorithms for on-line discovery of quantitative models without prior knowledge of the managed elements. Our two-phase approach is different in that we actively reconfigure the running system to try new parameters, rather than characterizing the system via passive sampling.

Another dimension in online optimization of a running system has been to assume *prior knowledge* about the system [7], rather than a black-box approach in our model. Menasce et al. [8] presented a system that performs online optimization of a web server by using hill-climbing techniques with a detailed knowledge of the performance of the server's components. Similarly, Rao et al. [9] and Lohman et al. [10] applied apriori system knowledge to automate configuration of databases in an online manner. These approaches are essentially different than ours as they require some prior knowledge of the target system being optimized and follow a white-box approach, whereas our two-phase approach require no prior knowledge of the system and strictly follows a black-box approach to keep generality of the problem being solved.

A crucial piece of optimizing a running system is the set of algorithms being used for optimizing system parameters. Evolutionary algorithms or heuristic search algorithms are typical approaches to optimizing a large-scale system such as networks with many parameters. The challenge is to search through large parameter spaces with minimal trials to get maximal information about the global solutions. Several realistic systems, such as network protocols in the Internet, can easily have parameter counts in the order of millions, a.k.a. curse of dimensionality. The design challenge has been to find the right balance between exploitation and exploration of parameter spaces. In this respect, hybrid designs have received a lot of attention from researchers. Such hybrid designs included merging of a genetic algorithm (GA) with a local search strategy based on the interior point method [11], using GA for global exploration and Ant Colony Optimization (ACO) for local exploitation [12], combining a calculus-based method with GA [13], mixing GA with Tabu Search [14] for solving resource scheduling problems [15], mixing ACO with Simulated Annealing for cluster analysis [16]. Our PTAS framework presents a new approach in using a search algorithm to find the right way of balancing the experiment budget among multiple algorithms.

Prior research on tuning link weights for interior gateway protocols (IGPs) is related to our work as well. In [17], the authors proposed a solution to set the IGP link weights according to the given network topology and traffic demand so as to control intra-domain traffic and meet TE objectives. The NP-hardness of the OSPF weight-setting problem is explained in [17]. In [18], they adopt objective of routing optimization as the minimization of the maximum link utilization in the network. In [19], they propose a hybrid genetic algorithm incorporating a local improvement procedure to the crossover operator of the genetic algorithm. The local improvement procedure makes use of an efficient dynamic shortest path algorithm to recompute shortest paths after the modification of link weights.

III. Two-Phase Online Network Optimization

Online optimization of systems can be performed in two ways: (i) using a separate model of the system for experimenting new configurations, (ii) using the system itself for experimentation without a separate system model.

In our previous work [2], we took the former approach which is illustrated in Figure 1. The framework included two cycles: optimization cycle and deployment cycle. The deployment cycle takes place at a time scale where it is possible to configure a new set of parameters in the real network system. The optimization cycle iterates through a model of the system, for which we used a network simulator, and attempts to find the best possible set of parameters to configure during the next occurrence of the deployment cycle. In that sense, the frequency of the deployment cycle is smaller than the optimization cycle. The underlying assumption in this approach is that the system-at-hand is stable and thus its behavior does not significantly change during at least one deployment cycle. This assumption is generally true for networks with a lot of fixed components such as backbone networks or wireless mesh networks.

However, this former approach fails when the network system is dynamic with high failure rates or a variable demand profile. For many practical networks, such as mobile ad-hoc networks (MANETs), the set of parameters which gives the optimum output metric may not give the optimum output after a short time. Thus, it is not practical to model such highly variant networks by simulations or other characterization techniques. To address management of such highly dynamic networks, we take the latter approach in this paper. We try to answer the key question of *"Is it possible and useful to use the real network itself as the model and try out new parameter configurations on the real network with the hope of finding better ones?"*

As illustrated in Figure 2, the optimization and deployment cycles are merged. We essentially perform in-situ trials of new configurations in the network itself to find better configurations. The key design issue is to search for better configurations without disturbing or temporarily deteriorating the network's existing performance. The intuitive motivation is that the network's performance may already be low since its dynamics are pretty much constantly changing. However, it is also possible that the network's performance might get noticeably disturbed when we are searching for better configurations.

Our approach employs two subsequent phases, *search* and *no-search*, where new configuration parameters are tried during the search phase. Each search phase is followed by a no-search phase with normal operation using the parameters found in the latest search phase. In this design, the following questions state some of the key tradeoffs:

- *How frequent the search should be done?*
- *How long should the search phase be?*
- *How worse the search phase can temporarily make the system performance due to its trials?*

In this work, we used an NS-2 simulator of a network to emulate a real-time running system and attempted to optimize the IGP link weights to obtain higher aggregate throughput. During the search phases, we used our black-box optimization algorithm, called Probabilistic Trans-Algorithmic Search (PTAS), to search for better IGP link weights. The PTAS framework can automatically find out the best set of algorithms that should work on the problem-at-hand. It is also adaptive to changes in the system behavior. This feature is especially useful for systems involving unexpected failures causing the response behavior to change, e.g., router or link failures in a network. We compared PTAS' performance against three other black-box optimization algorithms: Recursive Random Search (RRS) [20], Simulated Annealing (SA), and Genetic Algorithm (GA).

IV. PTAS: Probabilistic Trans-Algorithmic Search

The essence of PTAS is to apply an optimization search algorithm to find out how to best distribute available "experiment budget" to multiple optimization search algorithms. In other words, PTAS aims to systematize how to "search for the best search". First, we split our total budget into smaller chunks, and call them "round budgets". In the first round, PTAS allocates the round budget among the search algorithms equally. At the end of the first round, it compares the responses returned from each algorithm. When it allocates the round budget for the second round, it allocates the round budget among the algorithms based how they performed in the first round. When allocating the round budget for the third round, it allocates the round budget among the algorithms based how they performed in the second round, and so on. At the end of each round, PTAS transfers/exchanges the best result(s) among the algorithms.

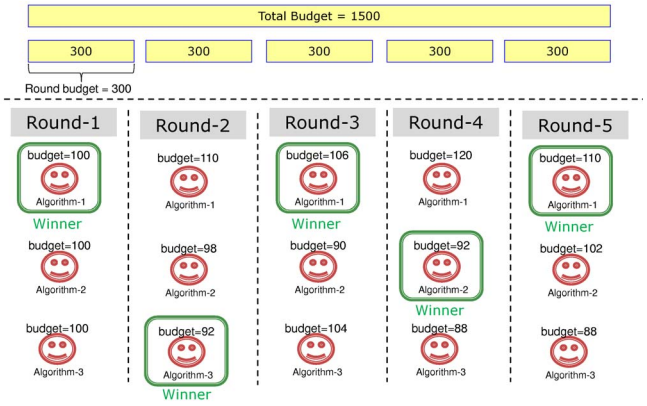


Fig. 3. A sample budget allocation scenario in five rounds.

The concept of "budget" models the amount of time allowed for PTAS to find a good solution. PTAS operates with two different budget scales:

- *Total Budget:* This is the total experiment budget that is available for PTAS. Each experiment (e.g., a simulation run or a real prototype experiment) is considered to take a certain amount of time.
- *Round Budget:* This is the budget that PTAS uses for each round which must be smaller than the Total Budget. The Total Budget typically will have several rounds in it. At each round, PTAS tunes and decides how to split the Round Budget across multiple search algorithms working for him. The budget each algorithm can get in a given round is based on how they performed compared to previous rounds.

Budget Allocator with Roulette Wheel: A crucial component of PTAS is to make the allocation of Round Budget to individual algorithms in an intelligent way so that the overall search performance is satisfactory. We use a "roulette wheel" method to determine how much to re-allocate the Round Budget among the individual algorithms.

Transfer of Best-So-Far Among Algorithms: PTAS employs multiple algorithms at each round. At the end of each round, PTAS gets the best result achieved by individual algorithms and transfers it into the other algorithms. In other words, if one of the algorithms found a very good sample in the previous round, the other algorithms also benefit from that sample by incorporating it into their search. How to make that incorporation might be different for each algorithm.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

V. Two-Phase Optimization of IGP Link Weights

To illustrate plausibility and efficiency of our two-phase approach, we apply our framework on the setting of Interior Gateway Protocol (IGP) link weights. This is a well-known intra-domain traffic engineering problem, and the goal is to make more efficient use of network resources within an autonomous system (AS). IGPs direct traffic based on link weights assigned by the network administrator. Each router in the AS computes shortest paths and creates destination tables used to direct each packet to the next router on the path to its final destination. Given a set of traffic demands between origin-destination pairs, the weight setting problem consists of determining weights to be assigned to the links so as to optimize a cost function, typically associated with a network congestion measure [19]. The NP-hardness of the IGP weight setting problem was shown before [17].

We used NS-2 simulations of a sample IGP to imitate a real network and evaluate the performance of the two-phase optimization using PTAS. We compared the performance of PTAS against the other three black-box optimization algorithms: Simulated Annealing (SA), Genetic Algorithm (SA), and Recursive Random Search (RRS). We used Exodus topology from the Rocketfuel dataset. There are 22 nodes and 37 links in the Exodus topology. We identified 18 of the 22 nodes as edge nodes and established $18 \times 17 = 306$ TCP flows between them. The goal is to maximize the aggregate throughput in the network, which is calculated as the total number of bytes received at sink nodes of the TCP flows. In our experiments, we picked 10 links to fail during the simulations, and ran the experiments 10 times for each link to gain confidence with a different seed.

In our previous work [2], initially, we randomly assign IGP link weights and run the simulation for a particular amount of time. This amount of time is what we call as "total budget", after which the metric to be optimized (e.g., the aggregate throughput) is sent to the PTAS. In the next round, PTAS changes the link weights and runs the simulation again to obtain a new metric value based on the new link weights. By repeating this iterative process for a pre-defined amount of time (i.e., deployment cycle in Figure 1), PTAS tries to come up with a set of link weights resulting in the optimum metric. This method is appropriate when the system is stable in its behavior, and thus the objective function does not change during a complete deployment cycle.

In this paper, we do not run the network simulator multiple times because we treat the simulator as the real running real system. We, again, define the optimization metric as the total number of bytes received at sink nodes of the TCP flows within a time interval. We call this time interval as "sampling interval" in our experiments. The number of metric samples to take is specified by "total budget". The simulator returns the metric, aggregate throughput, for the last sampling interval to the PTAS. Afterwards, PTAS changes the link weights in the topology. As a result of these link weight changes, the routing and the paths taken by the TCP flows change, and thus the aggregate throughput changes. This framework establishes a scheme of relationship between the sampling interval, the number of samples (i.e., total budget), and the search phase duration:

$$search\_phase\_duration = sampling\_interval * total\_budget$$

At the end of a search phase, the two-phase system deploys the best-found parameters which yield the optimum throughput. The system uses these link weights until the next search phase begins. We call the time between the search phases as "deployment phase", or no-search phase, and is related to the frequency of going into a search phase:

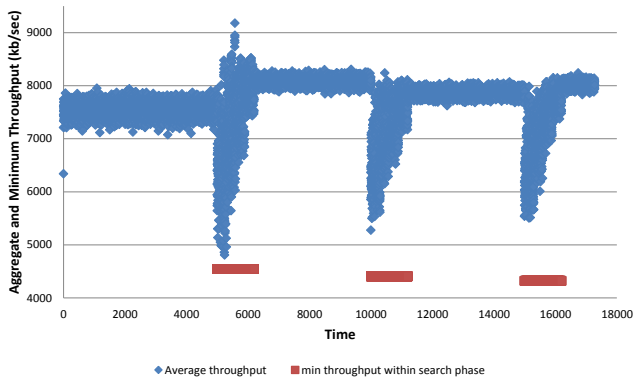
$$no\_search\_phase\_duration = 1/search\_frequency$$

Figure 4 illustrates aggregate throughput versus time for each algorithm when they are applied within our two-phase optimization framework. Total simulation duration is 17,280 seconds. The $no\_search\_phase\_duration$ is 5,000 seconds. In order to simulate an unstable and dynamic environment, we randomly fail links with an Exponentially distributed inter-failure time and failure duration. As can be observed, in some cases the throughputs after the search phases are worse than the throughputs before the search phases. The reason is that the environment is so unstable that link failures occur even during the search phases. We observed that PTAS handles the link failures well improves the throughputs, particularly when the failures occur during the search phases. The average of the throughputs for each of the sampling intervals shown in Figure 4 are 7,698.24, 7,322.22, 7,596.68, and 7708.21 for RRS, SA, GA, and PTAS, respectively.

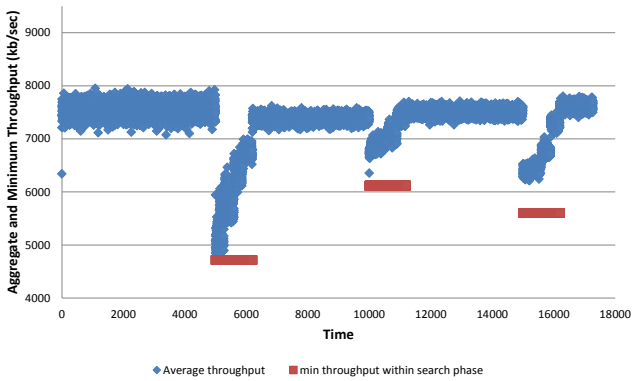
One important design tradeoff of our two-phase approach is to find an answer to the question: *How worse the search phase can temporarily make the system performance due to its trials?* It is possible that the network's performance might get disturbed when we are searching for better configurations. Figure 4 illustrates also how worse the search phase can temporarily make the system performance due to its trials. PTAS and RRS and achieving better throughput with a little bit lower minimum throughput values in comparison to GA and SA. This shows that GA and SA are more exploitive algorithms. PTAS and RRS perform well in finding a good balance between exploitation (i.e. higher average throughput) and exploration (i.e. lower minimum throughput); and they do not bring the average throughput to unreasonably low values while trying to maximize it.

We also compared the performance of the PTAS with other algorithms on different link failure rates. Figure 5 shows the performance of the algorithms for various link failure frequencies. We used two different values for the $number\_of\_rounds$ parameter of PTAS: 5 or 10. As expected, the aggregate throughput decreases when link failure frequency increases. Overall, PTAS performs similar to RRA but noticeably better than GA and SA. To gain confidence, we repeated the simulations 20 times with different seeds.

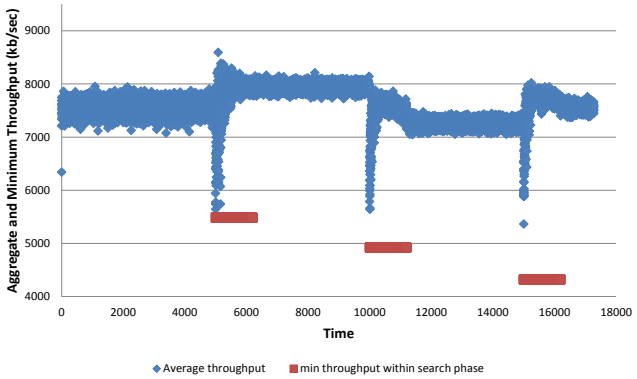
A key tradeoff to be observed in the system is *how frequently the system should go into search phase*. We used



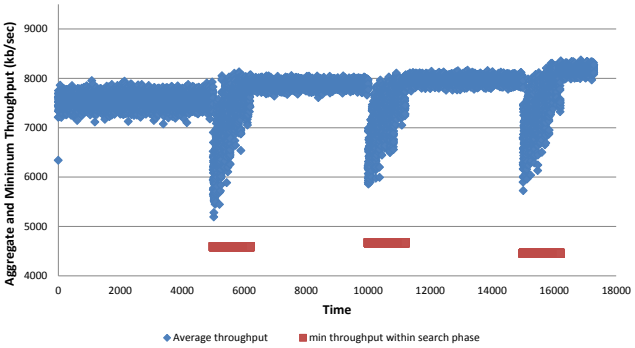
(a) Recursive Random Search (Average throughput: 7698.24)



(b) Simulated Annealing (Average throughput: 7322.22)



(c) Genetic Algorithm (Average throughput: 7596.68)



(d) PTAS (Average throughput: 7708.21)

Fig. 4. Comparison of PTAS with RRS, SA, and GA when running on real-time simulator

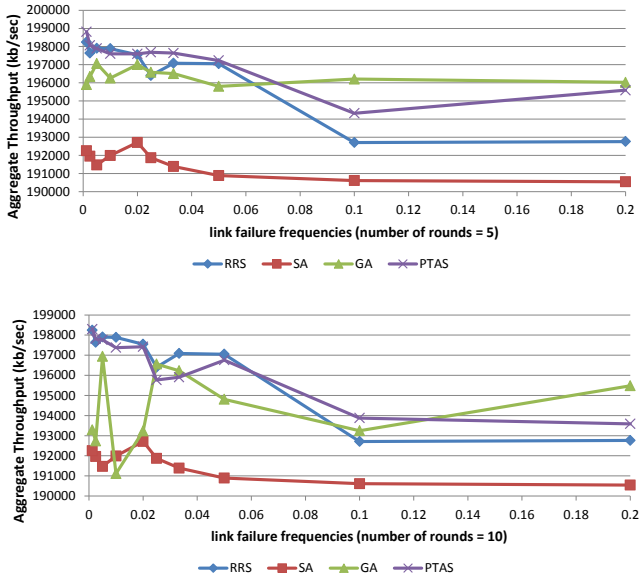


Fig. 5. Comparison of PTAS with RRS, SA, and GA over different link failure frequencies.

five different values of $no\_search\_phase\_duration$: 1000, 2000, 3000, 4000, and 5000. We observed that based on how dynamic the system is, there is an equilibrium point for PTAS. As we can see from Figure 6, going into the search phase in every 3000 time units seems to achieve the best results for PTAS in this particular case. Going into search phase less frequently reduces the aggregate throughput because the system is too dynamic and the optimum set of parameters found in one search phase may not be optimum until the next search phase, and going into another search phase more frequently than 1000 is needed. On the other hand, going into the search phase more frequently than 3000 reduces the aggregate throughput because according to "No Free Lunch" theorem, there is a cost for the search phase. Because this is a black-box search, for some of the parameters used, the system may result in low metric output, reducing the aggregate throughput.

The other key tradeoff to be examined is *how long a search phase should last*. That is, what should the $total\_budget$ for a search phase be? We used four different $total\_budget$ values: 300, 600, 900, and 1200. We observed that some algorithms improve the overall average throughput when they are given longer search phase intervals, while other algorithms degrade. Because the problem is a black-box problem and that the PTAS is a hybrid search algorithm with better adaptivity, it tends to outperform the other three algorithm regardless of the total budget. Lastly, a key parameter for PTAS itself is what the number of rounds should be for PTAS. That is, when PTAS is given a total budget to use in a search phase, into how many chunks the PTAS should divide this budget? We used four different values: 4, 6, 8, and 10. We observed that because

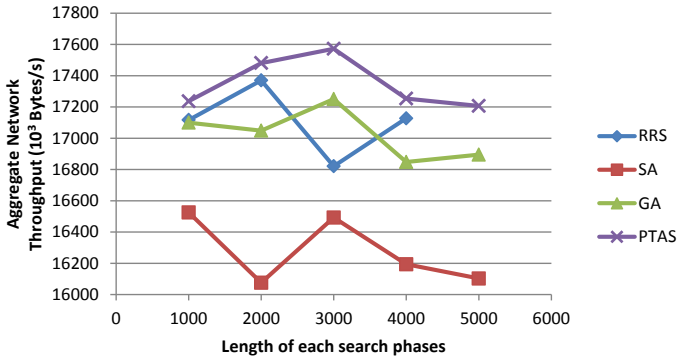


Fig. 6. Comparison of PTAS with RRS, SA, and GA over different subsequent search phase intervals

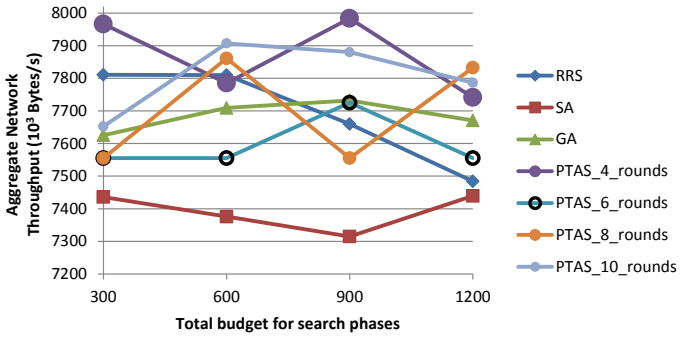


Fig. 7. Comparison of PTAS with RRS, SA, and GA for using different search phase lengths and different number of rounds for PTAS

of the instability of the system, PTAS tends to perform better when the number of rounds increases. This is also illustrated in Figure 7.

VI. Summary

Automated configuration and management of highly dynamic networks is a challenging problem for network practitioners. In this paper, we leverage our prior work, PTAS [2], on hybrid black-box optimization to adapt the search for changing network behavior. Using the system itself for experimentation without a separate system model, we perform in-situ trials in a network for optimization. We investigate a two-phase online optimization framework for network configuration and management. Our approach follows a pattern of two subsequent phases, *search* and *no-search*, where new configuration parameters are tried during the search phase. We explore some of the key tradeoffs in the two-phase model, such as how frequent the search should be done, how long should the search phase be, and how worse the search phase can temporarily make the system performance due to its trials. We apply the two-phase optimization model on the IGP link weight setting problem and observe the aggregate network throughput against several parameters such as link failure rate, optimization algorithm, and total experiment budget the search phases have. For such dynamic scenarios, we showed that PTAS outperforms the individual algorithms by a sizeable margin on average.

Acknowledgment

This work is supported in part by the US National Science Foundation awards 0721600 and 0721609.

References

[1] T. Ye, H. T. Kaur, S. Kalyanaraman, and M. Yuksel, "Large-scale network parameter configuration using an on-line simulation framework," *IEEE/ACM Tran. on Networking*, vol. 16, no. 4, pp. 777–790, 2008.

[2] B. Gonen, M. Yuksel, and S. Louis, "Probabilistic Trans-Algorithmic search for automated network management and configuration," in *IEEE International Workshop on Management of Emerging Networks and Services (IEEE MENS 2010)*, Miami, Florida, USA, 12 2010.

[3] X. Liu, L. Sha, Y. Diao, S. Froehlich, J. L. Hellerstein, and S. Parekh, "Online response time optimization of apache web server," in *Proceedings of the 11th international conference on Quality of service*, ser. IWQoS'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 461–478.

[4] P. V. Hentenryck and R. Bent, *Online Stochastic Combinatorial Optimization*. The MIT Press, 2009.

[5] C. Boutilier, D. C. Parkes, T. Sandholm, and W. E. Walsh, "Expressive banner ad auctions and model-based online optimization for clearing." in *AAAI'08*, 2008, pp. 30–37.

[6] Y. Diao, F. Eskesen, S. Froehlich, J. L. H. A. Keller, J. L. Hellerstein, A. Keller, L. F. Spainhower, and M. Surendra, "Generic on-line discovery of quantitative models for service level management," in *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management*. Kluwer Academic Publishers, 2003, pp. 157–170.

[7] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreement profits," in *Proceedings of the 3rd ACM conference on Electronic Commerce*, ser. EC '01. New York, NY, USA: ACM, 2001, pp. 213–223.

[8] D. A. Menascé, D. Barbará, and R. Dodge, "Preserving qos of e-commerce sites through self-tuning: a performance model approach," in *Proceedings of the 3rd ACM conference on Electronic Commerce*, ser. EC '01. New York, NY, USA: ACM, 2001, pp. 224–234.

[9] J. Rao, C. Zhang, N. Megiddo, and G. Lohman, "Automating physical database design in a parallel database," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '02. New York, NY, USA: ACM, 2002, pp. 558–569.

[10] G. M. Lohman and S. S. Lightstone, "Smart: Making db2 (more) autonomic," in *In VLDB 2002 28th International Conference on Very Large Data Bases , Kowloon Shangri-La Hotel, Hong Kong*, 2002.

[11] V. Kelner, F. Capitanescu, O. Lonard, and L. Wehenkel, "A hybrid optimization technique coupling an evolutionary and a local search algorithm," *J. of Computational and Applied Mathematics*, vol. 215, no. 2, pp. 448–456, 2008.

[12] Z.-J. Lee and C.-Y. Lee, "A hybrid search algorithm with heuristics for resource allocation problem," *Information Sciences*, vol. 173, no. 1-3, pp. 155–167, 2005.

[13] C.-T. Hsiao, G. Chahine, and N. Gumerov, "Application of a hybrid genetic/powell algorithm and a boundary element method to electrical impedance tomography," *J. of Computational Phy.*, vol. 173, no. 2, 2001.

[14] S. C. S. Porto and C. Ribeiro, "A tabu search approach to task scheduling on heterogeneous processors under precedence constraints," *International Journal of High-Speed Computing*, vol. 7, 1995.

[15] M. Zdansky and J. Pozivil, "Combination genetic/tabu search algorithm for hybrid flowshops optimization," *P. of ALGORITMY*, pp. 230–236, 2002.

[16] T. Niknam, B. B. Firouzi, and M. Nayeripour, "An efficient hybrid evolutionary algorithm for cluster analysis," *World Applied Sciences Journal*, vol. 4, 2008.

[17] B. Fortz and M. Thorup, "Increasing internet capacity using local search," *Comput. Optim. Appl.*, vol. 29, pp. 13–48, October 2004.

[18] A. Riedl, "A hybrid genetic algorithm for routing optimization in ip networks utilizing bandwidth and delay metrics," in *Proc. of IEEE Workshop on IP Operations and Management*, 2002, pp. 166–170.

[19] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup, "A hybrid genetic algorithm for the weight setting problem in OSPF-IS-IS routing," *Networks*, vol. 46, no. 1, pp. 36–56, 2005.

[20] T. Ye and S. Kalyanaraman, "A recursive random search algorithm for black-box optimization," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 3, pp. 44–53, December 2004.