Probabilistic Trans-Algorithmic Search for Automated Network Management and Configuration

Bilal Gonen, Murat Yuksel, and Sushil Louis
University of Nevada - Reno, Reno, NV 89557.
gonenb@cse.unr.edu, yuksem@cse.unr.edu, sushil@cse.unr.edu

***Abstract*—Online configuration of large-scale systems such as networks require parameter optimization to be done within a limited amount of time. This time limit is even more pressing when configuration is needed as a recovery response to a failure in the system. To quickly configure such systems in an online manner, we propose a Probabilistic Trans-Algorithmic Search (PTAS) framework which leverages multiple optimization search algorithms in an iterative manner. Essentially, PTAS applies a search algorithm to find out how to best distribute available experiment budget among multiple optimization search algorithms. Specifically, PTAS allocates experiment budget to each available search algorithm and observes each algorithm's performance on the system-at-hand. PTAS then probabilistically reallocates the experiment budget for the next round proportional to an algorithm's performance. This "roulette wheel" approach probabilistically favors the more successful algorithm in the next round. Following each round, the PTAS framework "transfers" the best found result(s) among the individual algorithms, making our framework "trans-algorithmic". PTAS thus aims to systematize how to "search for the best search". We show the performance of PTAS on well-known benchmark objective functions including scenarios where the objective function changes in the middle of the optimization process. To illustrate applicability of our framework to automated network management, we apply PTAS on the problem of optimizing link weights of an intra-domain routing protocol on a topology obtained from Rocketfuel dataset.**

***Index Terms*—black-box optimization; network management; network protocol configuration**

I. Introduction

Though there has been several tools and outcomes [1] from research on large-scale network management, network operators have found themselves more comfortable with trusting highly-experienced well-trained human administrators. However, the complexity of the management and configuration problem is increasing due to increasing heterogeneity in substrate technologies as well as applications demand for more stringent performance targets. Thus, tools to achieve automated ways of managing a running network are vitally needed. In this paper, we present a new black-box optimization algorithm for automated management and configuration of networks.

Global optimization is a commonly used technique for close-to-optimal configuration of systems, minimization of cost [2] or maximization of benefit [3]. One method of global optimization is known as "black-box" optimization, which considers the problem-at-hand as a black box and searches for optimal parameter setting yielding the best (i.e., minimum or maximum) output metric value. Figure 1 illustrates the black-box optimization approach to problem solving. As the black-box optimization is generic, the domain of applicable problems is vast as long as the output response of the problem system can be mapped to a single metric. The price of being generic and problem-independent comes typically as lack of optimality guarantees in the solution. Problem-specific methods like approximation algorithms [4], dynamic programming [5] or divide-and-conquer [6] are more successful in providing guarantees approaching the global optimum. However, these problem-specific methods may take long time to find a solution and typically require the whole search process to complete for a solution. Most of the real-time systems like an ISP network require parameter optimization to be done within a limited amount of time, e.g., in response to a network failure. Thus, the optimum solution may not be reachable for the real system-at-hand within the limited time budget.

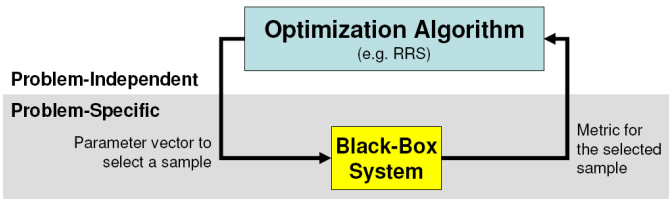


Fig. 1. Black-box optimization framework: Content of the black-box is problem-specific whilst the optimization algorithm can be generic.

The black-box optimization algorithms have to search for the optimal solution by exploring and then exploiting the response surface of the system. A critical design issue is to balance the amount of time spent on exploration and exploitation. Various black-box optimization search algorithms have been designed with different balancing of this tradeoff, e.g., Hill Climbing [7], Simulated Annealing [8], Genetic Algorithms [9], or Random Search [10]. Some of the search algorithms may perform better than other search algorithms on the same black-box problem depending on the response surface.

Automating a network's management and configuration can be quite difficult since reconfiguration of the entire network might be needed upon a major failure. Two critical aspects of the problem are: (i) network failures must responded very quickly and (ii) the network's optimal configuration can be quite different than the prior ones as the failures might have caused a significant change in the network's behavior (i.e., a major change in system response). To achieve fast

and adaptive network system configuration, we propose a Trans-Algorithmic Search (PTAS) framework which leverages multiple search algorithms in an iterative manner. PTAS allocates experiment budget (i.e., the number of experiments an algorithm can make to optimize the black-box system) to each available search algorithm and observes the success of them on the problem. Depending on their successes, PTAS reallocates the experiment budget for the next round. We make this reallocation based on a roulette wheel approach [11] favoring the more successful algorithm more in the next round. Following each round, the PTAS framework allows "transfer" of best found results among the algorithms being used, which makes our framework a "trans-algorithmic" one.

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 implement our hybrid search framework, by using three search algorithms, i.e., Recursive Random Search (RRS) [12], Simulated Annealing (SA), and Genetic Algorithm (GA). We show that PTAS can outperform any of the search algorithms on well-known benchmark objective functions. We also experiment with changing the objective function in the middle of the optimization process and show that PTAS outperforms a singleton algorithm more pronouncedly. Lastly, we apply PTAS on the problem of interior gateway protocol (IGP) link weights optimization on a realistic ISP topology and show that PTAS successfully finds intra-domain routing configuration providing better throughput.

The rest of the paper is organized as follows: In Section II, we cover prior work related to PTAS. In Section III, we detail our PTAS framework. In Section IV, we present our comparative performance of PTAS against three other heuristic algorithms. Section V presents application of PTAS on the IGP link weights optimization problem. We, then, summarize our work in Section VI.

II. Related Work

Numerous methods have been innovated to attack hard optimization problems in practical systems as well as to model them. From a systems management perspective, optimization means to determine the region of the parameter state space that leads to the "best" performance response. Typically, since the system's response surface is unknown, this question falls into the broad area of systematic, heuristic problem solving [13]. The main issues in problem solving involves developing an understanding of the difficulty of a problem: size of the search space, accuracy and ease of the evaluation function, and the nature of the problem constraints. In terms of being complete or partial solutions, techniques like Exhaustive Search, Local Search [14], Linear Programming [15] belong to the former group, and give useful, though sub-optimal results, even though the solution process is interrupted. There are several well-known techniques belonging to the latter group, such as greedy algorithms, divide-and-conquer strategies, Dynamic Programming [16] and A* algorithms [13].

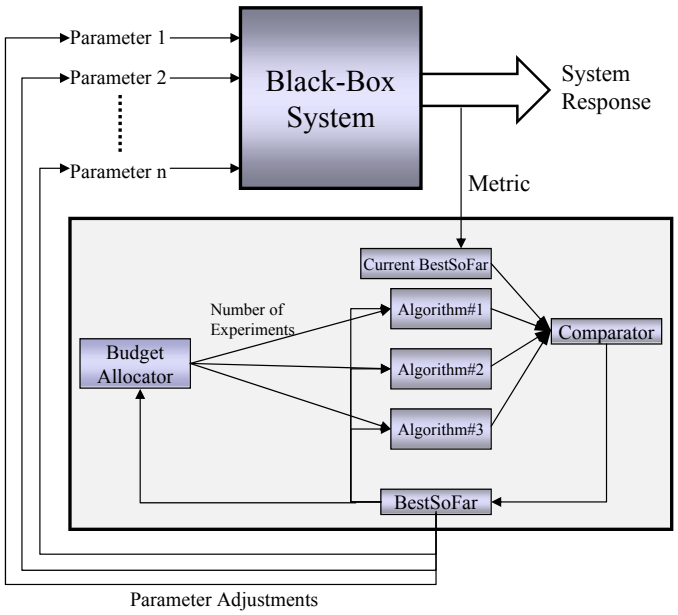


Fig. 2. Trans-Algorithmic Search for real-time system management: Each algorithm can experiment with the black-box system by trying out different parameter vectors.

A large set of techniques for complete solutions is the class of evolutionary algorithms [17] or heuristic search algorithms. In many cases, complex real systems have large parameter ranges which generally causes combinatorial explosion in the number of possible metric outcomes of such systems. The challenge is to search through such 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 [18], using GA for global exploration and Ant Colony Optimization (ACO) for local exploitation [19], combining a calculus-based method with GA [20], mixing GA with Tabu Search [21] for solving resource scheduling problems [22], mixing ACO with Simulated Annealing for cluster analysis [23]. In [24], authors develop a hybrid method combining Adaptive Partition-based Search (APS) [25] and Downhill Simplex Algorithm (DSA) [26], where APS is used for exploration and DSA for exploitation. Another hybridization was tried, in [27], between GA and a stochastic variant of simplex method [26].

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. The PTAS

design is generic and can work with any number of algorithms as long as the algorithms work with complete solutions.

Automated network management and configuration has been of high interest in the research and ISP communities. [1] In [28], the authors proposed a solution to set the link weights of interior gateway protocols (IGPs) according to the given network topology and traffic demand so as to control intra-domain traffic and meet traffic engineering objectives. The IGP link weight setting problem is known to be NP-hard [29]. In [30], the authors adopted objective of routing optimization as the minimization of the maximum link utilization in the network. Likewise, in [31], the authors proposed 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. The latest development on the IGP link weight setting problem is RRS [1], and we show that PTAS outperforms RRS on this critical problem.

III. 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, the budget allocator in the PTAS framework (Figure 2), allocates the round budget among the three 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 three 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 three algorithms based how they performed in the second round, and so on. At the end of each round, we transfer/exchange the best result(s) among the algorithms.

while $budget > 0$ **do**
  $best1 \Leftarrow Algorithm1(roundbudget1)$
  $best2 \Leftarrow Algorithm2(roundbudget2)$
  $best3 \Leftarrow Algorithm3(roundbudget3)$
  recalculate round budgets based on performances
  transfer bestOf(best1, best2, best3) to all three algorithms
end while

Total Budget: This is the total budget that we use for our experiment. To calculate the output of an objective function for some given parameters, we use eval function. Calling eval function one time costs one unit budget.

Budget for a round: This is the budget that we use for each round in our program. Initially we split this amount into three equally for three algorithms (RRS, SA, and GA). This budget does not change for each round; however the budget each algorithm can get in a given round is based on how they performed compared to previous round.

Total Budget: 1500
Round budget: 300

		Alg-1	Alg-2	Alg-3
Round-1	Round budget allocated	100	100	100
	Best result found	5000	4500	**4000**
	Best result 4000 is propagated among the algorithms.			
Round-2	Round budget allocated	80	100	120
	Best result found	4000	3950	**3800**
	Best result 3800 is propagated among the algorithms.			
Round-3	Round budget allocated	70	90	140
	Best result found	3700	**3200**	3500
	Best result 3200 is propagated among the algorithms.			
Round-4	Round budget allocated	60	110	130
	Best result found	3200	**2500**	2800
	Best result 2500 is propagated among the algorithms.			
Round-5	Round budget allocated	90	110	100
	Best result found	**1800**	2300	2500
	Total budget 1500 is used. Best result found is 1800			

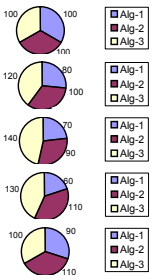


Fig. 3. Budget allocation in five rounds.

For our PTAS experiments in this paper, we used three algorithms as the individual algorithms available within the PTAS framework:

Recursive Random Search (RRS): The RRS algorithm is based on the initial high-efficiency property of random sampling and attempts to maintain this high-efficiency by constantly "restarting" random sampling with adjusted sample spaces. RRS algorithm uses random sampling for exploration and recursive random sampling for exploitation.

Simulated Annealing (SA): SA is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global optimum of a given function in a large search space.

Genetic Algorithm (GA): A GA is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A. Budget Allocator with Roulette Wheel

The crucial component of PTAS is to make the budget allocation to individual algorithms in an intelligent way so that the overall search performance is satisfactory. We use a "roulette wheel" [11] method to determine how much to reallocate the round budget among the individual algorithms. We call the PTAS component doing this budget distribution as "budget allocator" (see Figure 2).

The budget allocator first splits the round's budget into two portions: *hard* and *soft*. The hard round budget is the portion that has to be equally distributed among the individual algorithms, and the soft round budget is the portion that can be unequally distributed. The intuition behind this design is to give each algorithm a chance to perform in the next round. In this paper, we picked a soft round budget portion of 60%, which is a relatively conservative apportionment since only 60% of the whole round budget is reallocated each time. More aggressive budget allocation can be applied by increasing the soft round budget portion. Investigation of this matter is a future work.

The soft round budget is where the roulette wheel idea is applied. We distribute the soft round budget among the individual algorithms considering the amount of percentage

improvement they achieved in the previous round. Figure 3 illustrates a sample scenario how roulette wheel affects the budget allocation among the algorithms.

We now describe how to apply the roulette wheel technique on the soft round budget allocation. Let the round budget be $B = \beta_{RRS}[i] + \beta_{SA}[i] + \beta_{GA}[i]$, where $\beta_{RRS}[i]$, $\beta_{SA}[i]$, and $\beta_{GA}[i]$ are round i budget of RRS, SA, and GA respectively. Further, let the best-so-far value found during round i be $\Delta[i] = \min(\delta_{RRS}[i], \delta_{GA}[i], \delta_{GA}[i])$, where $\delta_{RRS}[i]$, $\delta_{GA}[i]$, and $\delta_{GA}[i]$ are the best-so-far value found during round i by RRS, SA, and GA respectively. We first calculate percentage improvement for each algorithm, as shown for RRS below:

$$T_{RRS} = \frac{\delta_{RRS}[i-1] - \delta_{RRS}[i]}{\delta_{RRS}[i-1]} \times 100$$

To calculate the round budgets for the next round, we multiply the current round budgets with fractions ($F_{RRS}[i]$, $F_{SA}[i]$, and $F_{GA}[i]$) for each algorithm:

$$\beta_{RRS}[i] = \beta_{RRS}[i-1] \times F_{RRS}[i]$$

The fractions are set to 1 initially. The expression below shows how to calculate fraction for RRS:

$$F_{RRS}[i] = F_{RRS}[i-1] + \frac{T_{RRS}[i]}{T[i]} \times 0.6 - \frac{0.6}{3}$$

where $T[i] = T_{RRS}[i] + T_{SS}[i] + T_{GA}[i]$.

For example, let's assume that RRS improved from $\delta_{RRS}[i-1]$=1000 to $\delta_{RRS}[i]$=800 during round i. Then the $T_{RRS}[i]$ is (1000-800)/1000*100 = 20. Further, let $T_{RRS}[i]$=20, $T_{SA}[i]$=10, and $T_{GA}[i]$=0. Also, let the round budgets in the current round be equal: $\beta_{RRS}[i]$=100 , $\beta_{SA}[i]$=100 , $\beta_{GA}[i]$=100.

$$F_{RRS}[i+1] = F_{RRS}[i] + (T_{RRS}[i]/T[i]) \times 0.6 - 0.6/3$$
$$F_{RRS}[i+1] = 1 + (20/30) \times 0.6 - 0.6/3 = 1.2$$
$$\beta_{RRS}[i+1] = \beta_{RRS}[i] \times F_{RRS}[i+1]$$
$$\beta_{RRS}[i+1] = 100 \times 1.2 = 120$$

In more simple terms, RRS gets back two third of the soft round budget 60*20/30 = 40, SA gets back one third 60*10/30 = 20. Since Because percentage improvement of GA is zero, GA gets back nothing 60*0/30 = 0. After this allocation, the round budgets become 120, 100, and 80 for RRS, SA, and GA respectively.

B. Transfer of Best-So-Far Among Algorithms

In PTAS, we run the three algorithms (RRS, SA, and GA) at each round. At the end of each round, we get best result of these three algorithms. If it is better than current best-so-far, we update the best-so-far sample (which is a parameter vector) and transfer it into these three 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.

For the RRS, in each round, we spend some of our budget for the exploration phase. Then we begin the exploitation phase around the point which has the best value found in the exploration phase. If we can not find any point better than best-so-far value in the exploration phase, then we begin the exploitation phase around the best-so-far. So, this is how we transfer the best-so-far sample to RRS algorithm.

For the SA, in the first round, we randomly pick a point as the initial state for the simulated annealing. After the first round, as the initial state of a round, we use the best result found in the previous round. This is how we transfer the best result of the three algorithms to the simulated annealing algorithm.

For the GA, in the first round, we generate samples (i.e., individuals) randomly to form our population. In each round, we find the worst individual in the population, replace it with best-so-far. So, this is how we transfer the best result of the three algorithms to genetic algorithm.

IV. Performance Evaluation

In order to compare our PTAS algorithm with other three individual search algorithms, we use several benchmark objective functions in the system. These are; Square Sum function, Rastrigin function, Griewangk's Function, Axis parallel hyper-ellipsoid function, Rotated hyper-ellipsoid function, and Ackley's Path function.

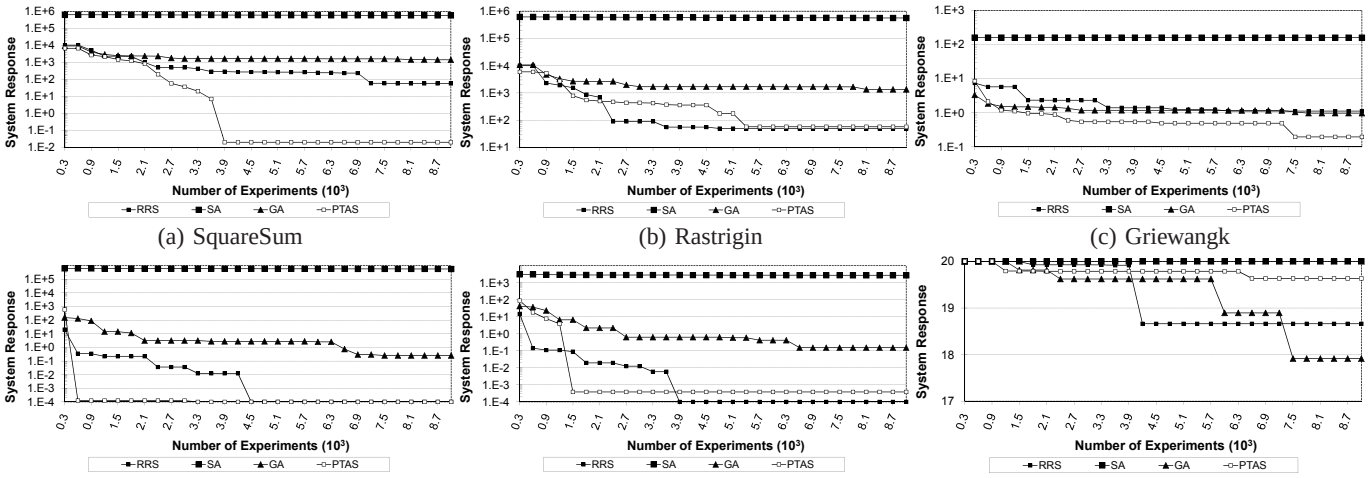
A. System Response is Fixed

Figure 4 shows a comparison of the four algorithms (i.e., RRS, SA, GA, and PTAS) when the objective function corresponding to the black-box system does not change. Number of experiments is 9000. The best-so-far values in the graph are average of five runs. We experimented with six different benchmark objective functions. For four out of the six different objective functions we tried, PTAS significantly outperformed the three individual algorithms by large margins. For the other two functions (i.e., Ackley and Rastrigin), PTAS was only slightly outperformed temporarily which suggests that fine tuning of PTAS can still result in better performance. Further, for those two functions, PTAS still performed better early on in the search (i.e., upto 3000 experiments), but was caught up by RRS later. This still makes PTAS more valuable, as the solutions are needed quickly for most real systems.

B. System Response Varies

In order to evaluate PTAS in a more realistic scenario, we change the objective function in the black-box during the optimization search process. This is a typical situation for large-scale real systems where parts of the system could temporarily fail and the system recovers from such failures. To imitate this, we change the objective function during the optimization search and return back to the original objective function.

We, again, compare PTAS with the three individual algorithms RRS, SA, and GA. The total number of experiments is 9000 and the best-so-far values we present are averaged over five runs. In each run, an original objective function is used for the first 3000 experiments. Then a temporary objective function is used for the second 3000 experiments, and finally,



(a) SquareSum (b) Rastrigin (c) Griewangk

(d) Axis parallel hyper-ellipsoid (e) Rotated hyper-ellipsoid (f) Ackley's Path

Fig. 4. Comparison of PTAS with RRS, SA, and GA for six benchmark objective functions.

the original objective function is used again for the third 3000 experiments.

Figure 5 shows the results when the original objective function is SquareSum and the temporary objective function is Rastrigin. Similarly, Figure 6 shows the results when the original objective function is Griewangk and the temporary objective function is Axis parallel hyper-ellipsoid. As both of the figures show, PTAS outperforms the individual algorithms significantly when system response changes during the optimization process. These results clearly show that PTAS can much better adapt to major changes in the system response by reallocating budget to better algorithm(s) after a change in the system response.

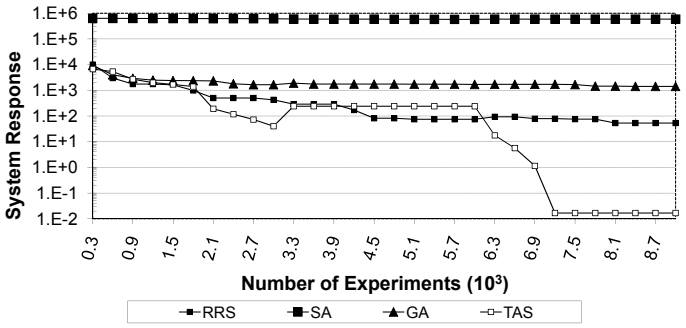


Fig. 5. Objective functions: SquareSum → Rastrigin → SquareSum

V. IGP Link Weights Optimization using PTAS

The goal of intra-domain traffic engineering is to make more efficient use of network resources within an autonomous system (AS). Interior Gateway Protocols (IGPs) direct traffic based on link weights assigned by the network operator. Each router in the AS computes shortest paths and creates destination tables, which are then used to direct 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

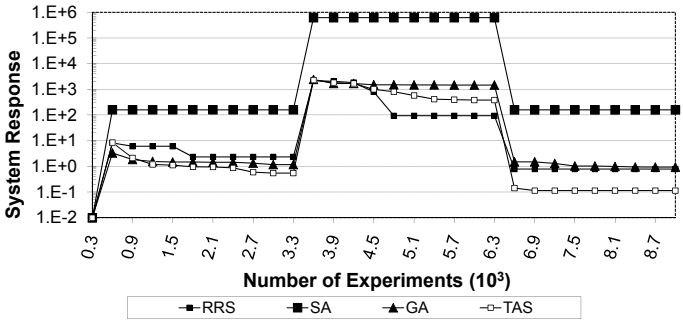


Fig. 6. Objective functions: Griewangk's → Rastrigin → Griewangk's

associated with a network congestion measure [31]. The OSPF weight-setting problem is shown to be NP-hard [29].

We mapped PTAS to a simulation-based IGP link weights optimization framework. PTAS selects a link weight value for each link in the simulated topology and feeds them to an NS-2 simulation of IGP routing. The metric to be optimized is the aggregate throughput of the network. Initially, we randomly assigned some weights between (0-100) to the links with a uniform distribution. Then, we ran the simulation with those link weights and let TCP traffic to flow for 100s. After each such simulation run, we measured aggregate throughput that was observed at the egress points of the network topology, and sent the throughput value as "the metric" to PTAS. In the next iteration, PTAS changes some of the link weights in the topology, and runs the simulation again. By repeating this iterative process up to an experiment budget, PTAS tries to find the IGP link weights closer to the optimum.

To model the network, we used Rocketfuel's Exodus topology, for which 22 nodes and 37 links exist. We used 7 nodes as the edge nodes, and composed 6×7=42 TCP flows between those edge nodes. As the simulation metric to be returned to the PTAS, we calculated total number of bytes received at sink nodes of the TCP flows. We repeated the optimization process 30 times to gain confidence. Figure 7 shows the average throughput achieved by each algorithm with

80% confidence intervals. We observe that algorithms improve their link weights as the experiment budget increases. PTAS outperforms the other three algorithms and its comparative performance becomes more pronounced as the experiment budget increases.

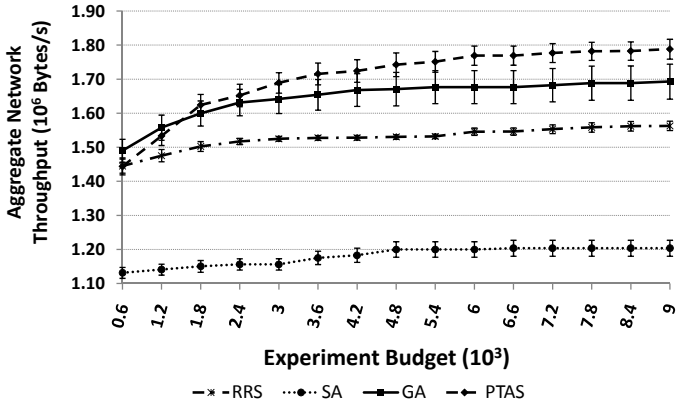


Fig. 7. Results of experiments on Exodus topology.

VI. Summary

Most of the search problems can be considered as black-box problems. Therefore, some of the search algorithms may perform better than other search algorithms on the same black-box problem. Because the response of such black-box problems to search algorithms is not foreseeable, we proposed a hybrid search framework, named "Trans-Algorithmic Search (PTAS)", for this kind of problems.

We implemented the PTAS framework by using three individual search algorithms. PTAS runs these algorithms in rounds. At the end of each round, PTAS changes the algorithms' experiment budgets based on how they performed in the previous round. By experimenting with benchmark objective functions, we compared our PTAS framework with the performance of individual algorithms. The results show that, on most of the objective functions, our PTAS framework performed significantly better than the individual algorithms. We also explored the scenarios when the objective function in the black-box changes during the optimization process. For such dynamic scenarios, we showed that PTAS outperforms the individual algorithms by a significant margin. We also applied PTAS on the well-known network configuration problem of IGP link weights setting and showed that PTAS outperforms the individual algorithms on that problem as well.

Acknowledgment

This work is supported in part by the U.S. 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] N. Ruana and X. Sun, "An exact algorithm for cost minimization in series reliability systems with multiple component choices," *Applied Mathematics and Computation*, vol. 181, no. 1, pp. 732–741, 2006.

[3] H. R. Liu, Y. F. Shen, Z. B. Zabinsky, C.-C. Liu, and S.-K. Joo, "Social welfare maximization in transmission enhancement considering network congestion," *IEEE T. on Pow. Sys.*, vol. 23, no. 3, pp. 1105–1114, 2008.

[4] V. V. Vazirani, *Approximation Algorithms*. Springer Verlag, 2002.

[5] M. Sniedovich, *Dynamic Programming*. CRC Press, 1992.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.

[7] F. Dehne, J.-R. Sack, and M. Smid, *Algorithms and Data Structures*. Springer, 1999.

[8] P. J. Laarhoven and E. H. Aarts, *Simulated Annealing: Theory and Applications*. Springer, 1987.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[10] A. A. Zhigljavsky and J. D. Pinter, *Theory of Global Random Search*. Springer, 1991.

[11] M. T. Jones, *AI Application Programming*. Charles River Media, 2003.

[12] 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.

[13] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Springer-Verlag, 2000.

[14] P. Pardalos, A. Migdalas, and R. E. Burkard, "Combinatorial and global optimization," *Series on Applied Mathematics*, vol. 14, pp. 55–74, 2002.

[15] B. Feiring, *Linear Programming*. Sage Publications, 1986.

[16] R. Bellman, *Dynamic Programming*. Dover Publications, 2003.

[17] T. Baeck, D. Fogel, Z. Michalewicz, and T. Back, *Evolutionary Computation 1: Basic Algorithms and Operators*. Inst. for Physics, 2000.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] J. A. Szab, "An efficient hybrid optimization procedure of adaptive partition-based search and downhill simplex methods for calibrating water resources models," *Geophysical Research Abstracts*, vol. 10, 2008.

[25] B. I. Kumova, "Dynamically adaptive partition-based data distribution management," *Proc. of Workshop on Principles of Advanced and Distributed Simulation*, pp. 292–300, 2005.

[26] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, 1965.

[27] J. Yen, D. Randolph, J. C. Liao, and B. Lee, "A hybrid approach to modeling metabolic systems using genetic algorithm and simplex method," *P. of Conf. on AI for Apps.*, vol. 20-23, pp. 277–283, 1995.

[28] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing ospf weights," *Proceedings of the INFOCOM 2000*, pp. 519–528, 2000.

[29] ——, "Increasing internet capacity using local search," *IEEE Transaction on Networking*, 2000.

[30] 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.

[31] 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.