

# Parallel Recombinative Simulated Annealing for Wavelength Division Multiplexing

Dale R. Thompson and Md. Tanveer Anwar  
Department of Computer Science and Computer Engineering  
University of Arkansas  
Fayetteville, Arkansas 72701

*Abstract— Parallel recombinative simulated annealing (PRSA) is applied to the static lightwave establishment network operation problem for wavelength division multiplexing (WDM) networks. It is assumed that the cross-connects do not perform wavelength conversion. An asynchronous version of PRSA is implemented on a multiprocessor computer with multiple interacting populations. The cooling coefficient, population size, and the number of migrants is varied for PRSA to determine the affect on the convergence rate and the quality of the final solution. It behaves like a simulated annealing (SA) algorithm, but has the advantage of being easy to implement on multiple processors.*

*Keywords— Genetic algorithms, simulated annealing, parallel computing, wavelength division multiplexing.*

## I. INTRODUCTION

Wavelength division multiplexing (WDM) is a technology that enables multiple optical signals over a single fiber by carrying signals on separate wavelengths [1]. Virtual topologies can be mapped onto the physical topology because of the multiple wavelengths [2]. For example, a single optical virtual channel consisting of one wavelength can be established between two nodes on the network to transfer information. The two nodes appear to be directly connected even though the virtual channel may pass transparently through several other nodes. Therefore, all-optical virtual channels between nodes that require large bandwidths and/or low delays can be established. The set of these wavelengths is called the virtual topology.

The design and performance analysis of a WDM network consists of two problems [3]. First, is the network design problem. Given the traffic demand and possibly a partial physical topology the problem is to find the number of fibers, the placement of optical amplifiers, the placement of cross-connects, and the size of cross-connects. Typically, the measure of the quality of the solution in the network design prob-

lem is cost. Second, is the network operation problem. Given the traffic demand and the complete topology the problem is to find the routing and wavelength assignments. The objective in the network operation problem is to minimize the cost, to maximize the carried traffic, or to minimize the probability of blocking at a specified traffic utilization.

At each node there are WDM cross-connects. There are two general categories of WDM cross-connects [3]. Wavelength selective cross-connects (WSXC) transfer wavelengths from input ports to output ports with no wavelength conversion. The original wavelength remains constant. Wavelength interchanging cross-connects (WIXC) transfer wavelengths and may convert a wavelength from an input port to a different wavelength before transferring it to the output port. This is desirable because there are a limited number of wavelengths that can be supported on a fiber and the incoming wavelength may be the same as an existing wavelength on the outgoing port.

If a network only uses WSXCs that do not perform wavelength conversion, a wavelength may not be able to be assigned between the source and destination even though there are available wavelengths on all links [4]. If the wavelengths are fixed, instead of dynamically assigned, then this problem is referred to as the static lightwave establishment (SLE) WSXC network operation problem. This is a combinatorial problem with several local minima [2]. Minimizing the number of wavelengths can be formulated as an integer problem. Maximizing the carried traffic can be formulated as a multicommodity maximum-flow problem. But, neither formulation scales well to large networks [3]. Therefore, approximate heuristic methods are typically used for solving the SLE WSXC network operation problem such as the Longer Paths First algorithm [3], simulated annealing [2], [5], stochastic rulers [6], or genetic algorithms [7]. Given a set of traffic flows these techniques are applied to determine good routing and wavelength assignments within a reasonable amount of computational time.

Parallel recombinative simulated annealing (PRSA) combines the strengths of a population-based genetic algorithm (GA) with the well-understood convergence properties of simulated annealing (SA) [8]. PRSA has been applied to combinatorial problems such as cell placement in VLSI design [9] and image segmentation [10]. In this paper, we apply PRSA to the SLE WSXC network operation problem. We vary the parameters of PRSA such as cooling coefficient, population size, and number of migrants and find it to be a robust technique.

## II. PARALLEL RECOMBINATIVE SIMULATED ANNEALING

The PRSA algorithm combines GA and SA into a hybrid parallel algorithm [8]. It inherits the convergence properties of SA to help GA converge. This convergence is introduced by holding competitions between the parents and the children using the Boltzmann or Metropolis criterion. In this paper, we use the Metropolis criterion. Like GA, PRSA is relatively easy to implement on a multiprocessor computer by running populations on different processors. In this section, we introduce GA, SA, and PRSA.

### A. Genetic Algorithm

GA is a heuristic optimization technique for approximating the global solution to a complex problem space. It is a good general strategy for real-world problems where objective functions and constraints are analytic [11]. It is based on “natural selection” of competing solutions and “genetic” encoding of each of those solutions. GA uses reproduction, mutation, competition, and selection processes. It begins with a population of individuals representing solutions that are often generated randomly. A fitness value is assigned to each individual. Two individuals are chosen to reproduce and exchange genetic material to create offspring. A crossover operator determines how the genetic material is exchanged. Mutation of individuals is introduced to add genetic information that was lost or did not exist in the initial population. Competition for the finite number of population slots is accomplished by a selection process that is biased in favor of better individuals. These steps are repeated for several generations to produce populations with more fit solutions [12]. GA is good at coarsely exploring the search space but is poor at precisely finding the local minima in the region that it finally selects. Therefore, the combination of a simple local search algorithm with GA is common [13].

### B. Simulated Annealing

SA is a heuristic optimization technique based on the physical process of crystallization [14]. Assume a minimization problem. Given a current solution  $x$  of cost  $f(x)$  a local search method finds a new solution  $y$  of cost  $f(y)$ . If the new solution costs less than the current solution, it replaces the current solution. If it costs more than the current solution, it replaces the current solution with probability  $\exp[-(f(y) - f(x))/T]$ . The temperature control parameter  $T$  controls the probability of accepting a solution that has a higher cost than the current solution, which permits SA to escape local minima. In SA, the initial temperature is high causing it to behave like a random search algorithm. Then, the temperature is decreased according to a cooling schedule and a local search method is applied again. As the temperature is decreased the probability of accepting a higher cost solution decreases. As the temperature approaches zero, the probability of accepting a higher cost solution goes to zero. The initial temperature and the cooling schedule determine the quality of the solution. A slower cooling schedule results in higher quality solutions, but increases the number of evaluations [15]. In theory, SA with a logarithmic cooling schedule converges with probability one to the global optimum solution, but requires an infinite number of iterations [16]. A common practice is to use a geometric cooling schedule of  $T \leftarrow cT$  where  $c < 1.0$  with values of  $c$  between 0.80 and 0.99 [17].

### C. PRSA

The PRSA algorithm for asynchronous operation on multiple processors with a different population on each processor is shown in Fig. 1. A serial version and synchronous parallel version are in [8]. We chose this version of PRSA because we wanted to harness the CPU power of a multiprocessor machine for decreased wall-clock time and we wanted to experiment with the performance of PRSA for varying number of migrants that were sent and received between populations.

The Metropolis criterion in the algorithm of Fig. 1 is based on the Metropolis sampling procedure for a target distribution  $\pi(x)$  and can be represented by (1) [14].

$$\min \left( 1, \frac{\pi(y)}{\pi(x)} \right) \quad (1)$$

where  $x$  is the current solution,  $y$  is the candidate solution,  $\pi(x)$  is the probability of being in state  $x$  of the form  $\pi \propto \exp\left(\frac{-f(x)}{T}\right)$ , and  $T$  is a parameter

---

```

 $T \leftarrow T_{initial}$ 
Initialize population of  $n$  members
Repeat  $g$  times
  Do  $n/2$  times
    Select 2 parents at random
    Generate 2 children using crossover and
      mutation
    Hold one or two competitions using the
      Metropolis criterion between children
      and parents
    Overwrite parents with trial winner
  end do
Periodically lower  $T$ 
Send/receive migrants to/from other
processors

```

---

Fig. 1. PRSA algorithm

known as temperature.

The probability in (1) is implemented by accepting the candidate solution if its fitness is less than the current solution. If the fitness of the candidate solution is greater than the current solution, it is accepted with probability  $\exp(-\Delta E/T)$ , where  $\Delta E = f_{candidate} - f_{current}$ . The temperature parameter  $T$  begins at a high temperature and is slowly lowered as in SA. The control parameter  $T$  controls how sharply  $\pi(x)$  is peaked at the minima of fitness function  $f$  [18].

Note that there are several variations for implementing the Metropolis criterion in PRSA [8]. Both parents could compete as a unit against the children. Or each individual child could compete against one of the parents. In our implementation of PRSA each child competes with a different parent.

In the asynchronous version of PRSA, each population sends and receives migrants to and from other populations. The number of migrants and which populations communicate are parameters. In this paper, the populations are arranged in a ring and a population always receives migrants from its left neighbor and sends migrants to its right neighbor. The number of migrants is held constant for a run, but we vary the number of migrants as a parameter to determine the affect on the rate of convergence and the quality of the final solution.

### III. PROBLEM FORMULATION

We define a traffic parcel as the number of wavelengths required between each source and destina-

tion on the WDM network. PRSA chooses the set of routes for *all* traffic parcels to minimize the *total* wavelength cost. The cost of a route is based on the distance between the source and destination. The candidate routes for each traffic parcel are calculated using a  $k$ -shortest path algorithm by Eppstein [19]. The  $k$ -shortest path algorithm finds the shortest, second shortest, up to the  $k$ -shortest route in the network. Then, the wavelengths along the routes are assigned using the *First-Fit* heuristic [3]. All wavelengths on a link are indexed and a wavelength is assigned on a route by searching through the wavelengths until it finds the first wavelength that can be assigned on all links. We also require that each wavelength has a backup wavelength that should be assigned on an alternate route. PRSA penalizes a backup wavelength that is assigned on the same route as the primary wavelength, but does permit such configurations.

In PRSA, like GA, a chromosome represents a solution and consists of a string of “genes” with each gene representing a variable in the solution. For the SLE WSXC network operation problem the chromosome consists of two rows of integers as seen in Fig. 2. Each gene is an integer between 0 and  $k - 1$  representing the shortest route up to the  $k$ -shortest route. Each column represents a traffic parcel for a particular source-destination pair. The first row represents the primary route for each traffic parcel and the second row represents the backup route. In Fig. 2 there are fifteen traffic parcels and the three shortest routes indexed 0 to 2 are candidate routes.

0	1	0	2	1	2	1	1	0	1	2	0	0	0	1
1	2	1	0	1	1	0	0	2	2	1	1	1	2	0

Fig. 2. Chromosome representing routing assignments for fifteen traffic parcels

Less desirable solutions are penalized instead of rejected. First, a backup wavelength that is assigned on the same route as the primary wavelength is penalized by raising the route cost to 1.5. This is necessary for a node that does not have an alternate route to another node. Second, a wavelength that is assigned and exceeds the maximum number of wavelengths on a link is penalized by raising the link cost to 1.5. These penalties appear to work better than simply rejecting these configurations.

A set of random physical topologies on which to map the wavelengths was generated with the topology generator *gt-itm* [20]. The Waxman model was used to generate random physical topologies that are similar

to real network topologies. Although twenty random topologies were generated, results on only one of the topologies is presented to compare the performance of PRSA with varying parameters.

The number of required wavelengths between each source and destination was generated randomly with a uniform distribution in the range  $[1, 14]$  such that the total number of wavelengths  $\lambda_T$  was greater than an upper bound on the total number of wavelengths that could be carried by the physical topology  $\lambda_{max}$ . Each traffic parcel was then scaled to present PRSA with a difficult task of assigning wavelengths. First, an estimate of  $\lambda_{max}$  was calculated by multiplying the number of links by the maximum number of wavelengths that each link could carry. This calculation assumed that each route had only one link, which was not the case. So this is an upper bound on  $\lambda_{max}$ . For  $\lambda_T > \lambda_{max}$ ,  $\lambda_T$  was scaled so that it was 50% of  $\lambda_{max}$ . Each traffic parcel was multiplied by  $(0.5)\lambda_{max}/\lambda_T/2$ , where the extra  $1/2$  accounts for each traffic parcel having an alternate route. This proved to provide a challenging problem for PRSA because the average route has greater than one route. So even at 50% of  $\lambda_{max}$  only about 50% of the wavelengths could be assigned with the new scaled offered traffic.

Like SA, the quality of the solution for PRSA is dependent on the cooling schedule. The cooling schedule consists of the initial temperature  $T_i$ , final temperature  $T_f$ , and the method used to decrease the temperature. The method from [8] was used to set  $T_i$  and  $T_f$ . The initial temperature was set such that  $\Delta E = \bar{f} + \sigma$ , where  $\bar{f}$  is the average fitness and  $\sigma$  is the standard deviation of the initial population that is generated randomly. The probability of accepting a worse solution  $p$  is set to 0.25. Therefore, (2) is solved for  $T$  to find  $T_i$ .

$$p = \exp(-\Delta E/T) \quad (2)$$

The final temperature  $T_f$  is calculated by determining the maximum energy difference  $\Delta E$  to detect and setting the probability of accepting a more expensive solution to a low value. PRSA was run with a near-zero  $T_f$  with different random seeds and  $\Delta E$  was set to the difference between the best and second best solutions found. Using this  $\Delta E$  and setting  $p = 0.01$  (2) is solved for  $T$  to find  $T_f$ .

The geometric cooling schedule  $T \leftarrow cT$  was used. Using this cooling schedule then  $T_f$  can be calculated with (3)

$$T_f = T_i c^{g/s} \quad (3)$$

where  $c$  is the cooling coefficient  $c < 1$ ,  $s$  is how many generations to run before lowering  $T$ , and  $g$  is the number of generations that PRSA is run. Given  $T_i$ ,  $T_f$ ,  $s$ , and  $c$  the number of generations  $g$  required to reach  $T_f$  is calculated. The higher the value of  $c$  corresponds to a slower cooling schedule.

#### IV. NUMERICAL RESULTS

A 25-node network with average node degree 3.0 was generated with *gt-itm* [20] using the Waxman model and is shown in Fig. 3. The three shortest paths were used as candidate routes for assignment by PRSA and were indexed from 0 to 2 with 0 being the shortest path. The offered traffic was generated randomly as previously discussed and scaled to 50% of the absolute maximum number of wavelengths that the physical topology could carry. There were 600 traffic parcels ( $25 \times (25 - 1)$ ). Therefore, there were 600 columns in the chromosome. Each link consisted of two one-way fibers that could support a maximum of 50 wavelengths on each fiber. PRSA was run on a multiprocessor computer using four of the processors and the Message Passing Interface (MPI) to communicate between populations. Using the method discussed in the previous section  $T_i = 1.08 \times 10^7$  and  $T_f = 22$ . But, we ran PRSA until the temperature was significantly lower than  $T_f = 22$  to ensure a converged state.

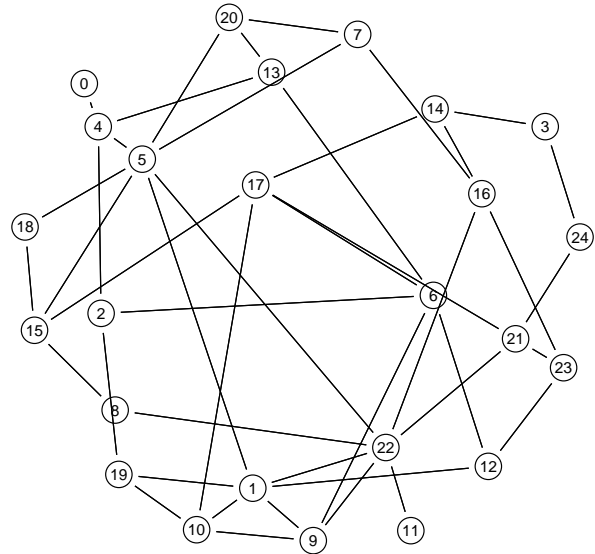


Fig. 3. Physical topology

First, the cooling coefficient  $c$  was varied to determine the affect on the convergence rate and the quality of the final solution. With the population size  $n = 50$  and the number of migrants sent and

received  $m = 2$ , PRSA was run with cooling coefficients  $c \in \{0.80, 0.90, 0.99\}$ . The initial temperature is the same, but the final temperature is different for each of the test cases. For lower values of  $c$  the final temperature is less. The best performing population of the four processors was chosen to compare the different cooling coefficients. The average fitness of the population is plotted versus the number generations for each cooling coefficient in Fig. 4.

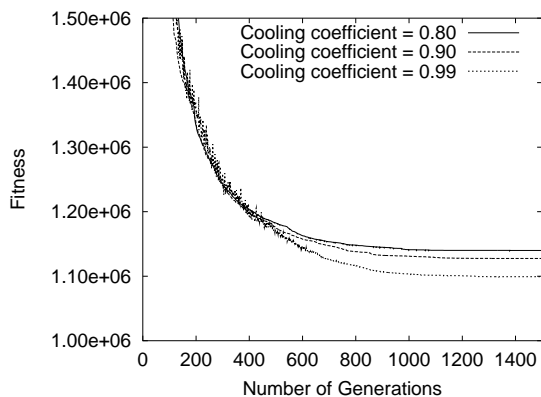


Fig. 4. Average fitness vs. number of generations for varying cooling coefficients

The convergence rate is quicker for the smaller cooling coefficients at the expense of the quality of the final solution. This is expected since a slower cooling schedule in SA results in higher quality solutions.

Then, the population size was varied to determine the affect on the convergence rate and the quality of the final solution. With  $c = 0.99$  and  $m = 2$ , the parameters  $g$  and  $s$  were set so that the same  $T_f$  was reached with the same number of function evaluations for population sizes  $n \in \{10, 25, 50\}$ . For example, with  $n = 50$  PRSA was run with  $g = 2500$  and  $s = 1$  while for  $n = 10$  PRSA was run with  $g = 12500$  and  $s = 5$ . The best performing population of the four processors was chosen to compare the different population sizes. The best fitness of the population is plotted versus the number of function evaluations per processor for each population size in Fig. 5.

The convergence rate is quicker for the smaller population sizes. PRSA converges quicker for  $n = 10$ , but the cost of the final solution is higher than with  $n = 25$  or  $n = 50$ . For  $n = 25$  and  $n = 50$  PRSA converges to approximately the same cost solution, although for  $n = 25$  the convergence rate is quicker. A larger population size provides a larger sampling of the solution space giving a higher quality solution at the end. But it also decreases the rate of convergence to the final solution found. So there is a trade-off be-

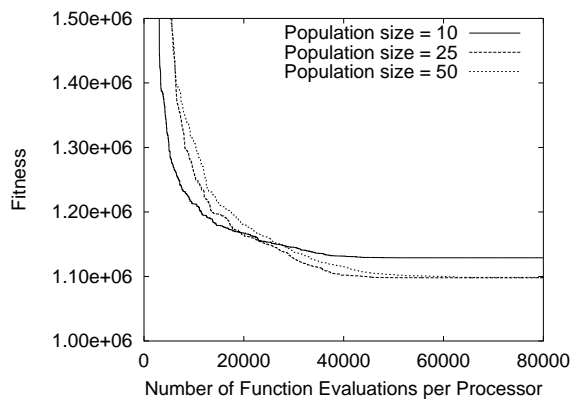


Fig. 5. Best fitness vs. number of function evaluations per processor for varying population size

tween the convergence rate and the quality of the final solution for different population sizes. Notice that for both  $n = 25$  and  $n = 50$  there is no significant difference between the costs of the final solutions. So for this particular problem  $n = 25$  would be a better choice.

Finally, the number of migrants sent and received  $m$  was varied to determine the affect on the convergence rate and the quality of the final solution. With  $n = 50$  and  $c = 0.99$ , PRSA was run with  $m \in \{2, 6, 10\}$ . The best performing population of the four processors was chosen to compare the different values of  $m$ . The average fitness of the population is plotted versus the number generations for each  $m$  in Fig. 6.

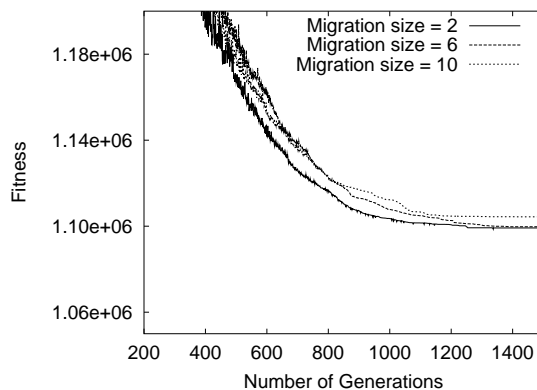


Fig. 6. Average fitness vs. number of generations for varying migration sizes

For smaller  $m$ , PRSA converges quicker and to a lower cost solution. For  $m = 2$ , PRSA converges quicker than  $m = 6$  and both reach a solution that has approximately the same cost. But, for  $m = 10$  the convergence rate is slower than for  $m = 2$  or  $m = 6$  and the final solution costs more than either.

If too many solutions are exchanged, then the

lower cost solutions during the first few generations may overtake the other populations and not permit them to explore the solution space for better solutions. This can also be explained by the work of Azencott on parallel SA algorithms in [21]. Azencott proved that simultaneous periodically interacting searches performed with a common cooling schedule are asymptotically *less efficient* than performing independent searches [21]. In other words, it is better to perform independent noninteracting searches and choose the best answer from among the independent searches. This seems to be supported by our results.

## V. CONCLUSIONS

In this paper, we applied PRSA to the SLE WSXC network operation problem in WDM networks. PRSA is presented as another method to solve a combinatorial problem that combines the strengths of a population-based GA with the well-understood convergence properties of SA. We varied the parameters of PRSA including the cooling coefficient, the population size, and the number of migrants. Like SA, a smaller cooling coefficient that causes a faster decrease in temperature increases the convergence rate at the expense of the final solution. A larger population size provides a larger sampling of the solution space increasing the quality of the final solution, but decreases the convergence rate. Note that increasing the population size has diminishing benefits.

An interesting result of this work is the affect of varying the number of migrants that were exchanged between the populations. If the best result is chosen from among the separate populations, a lower number of migrants performed better than a greater number of migrants. A lower number of migrants increased the convergence rate *and* improved the quality of the final solution. This seems to contradict the expected result, but has theoretical support from [21]. We have a possible modification to improve these results that will be reported in the future.

## REFERENCES

- [1] D. Cavendish, "Evolution of optical transport technologies: from SONET/SDH to WDM," *IEEE Communications Magazine*, vol. 38, no. 6, pp. 164–172, 2000.
- [2] J. A. Bannister, L. Fratta, and M. Gerla, "Designing metropolitan area networks for high-performance applications," *Computer Networks and ISDN Systems*, vol. 2, pp. 223–230, 1990.
- [3] E. Karasan and E. Ayanoglu, "Performance of WDM transport networks," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1081–1096, 1998.
- [4] E. Karasan and E. Ayanoglu, "Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 186–196, 1998.
- [5] B. Mukherjee, D. Banerjee, S. Ramamurthy, and A. Mukherjee, "Some principles for designing a wide-area WDM optical network," *IEEE/ACM Transactions on Networking*, vol. 4, no. 5, pp. 684–696, 1996.
- [6] A. Ganz and X. Wang, "Efficient algorithm for virtual topology design in multihop lightwave networks," *IEEE/ACM Transactions on Networking*, vol. 2, no. 3, pp. 217–225, 1994.
- [7] D. Saha, M. D. Purkayastha, and A. Mukherjee, "An approach to wide area WDM optical network design using genetic algorithm," *Computer Communications*, vol. 22, pp. 156–172, 1999.
- [8] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: a genetic algorithm," *Parallel Computing*, vol. 21, no. 1, pp. 1–28, 1995.
- [9] K. Kurbel, B. Schneider, and K. Singh, "Solving optimization problems by parallel recombinative simulated annealing on a parallel computer - An application to standard cell placement in VLSI design," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 28, no. 3, pp. 454–461, 1998.
- [10] S. M. Bhandarkar and H. Zhang, "Image segmentation using evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 1–21, 1999.
- [11] T. Bäck, U. Hammel, and H. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.
- [12] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.
- [13] P. Jog, J. Y. Suh, and D. Van Gucht, "The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem," in *Proceedings of the 3rd International Conference on Genetic Algorithms*. 1989, pp. 110–115, San Mateo, CA: Morgan Kaufmann.
- [14] S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [15] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, New York, NY: Wiley, 1989.
- [16] F. Romeo and A. Sangiovanni-Vincentelli, "A theoretical framework for simulated annealing," *Algorithmica*, vol. 6, no. 5, pp. 302–345, 1991.
- [17] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, New York, NY: Halsted Press, 1993.
- [18] G. L. Bilbro, "Fast stochastic global optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 684–689, 1994.
- [19] D. Eppstein, "Finding the k shortest paths," *SIAM J. Computing*, vol. 28, no. 2, pp. 652–673, 1998.
- [20] E. Zegura, K. Calvert, and M. Donahoo, "A quantitative comparison of graph-based models for internet topology," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 770–783, 1997.
- [21] R. Azencott, *Simulated annealing: parallelization techniques*, New York, NY: John Wiley & Sons, 1992.

D. R. Thompson and M. T. Anwar, "Parallel recombinative simulated annealing for wavelength division multiplexing," in *Proceedings of the 2003 International Conference on Communications in Computing*, pp. 212-217, Las Vegas, NV, June 23-26, 2003.