BIASED RANDOM-KEY GENETIC ALGORITHMS WITH APPLICATIONS IN TELECOMMUNICATIONS

MAURICIO G.C. RESENDE

Abstract. This paper surveys several applications of biased random-key genetic algorithms (BRKGA) in optimization problems that arise in telecommunications. We first review the basic concepts of BRKGA. This is followed by a description of BRKGA-based heuristics for routing in IP networks, design of survivable IP networks, redundant server location for content distribution, regenerator location in optical networks, and routing and wavelength assignment in optical networks.

1. Introduction

Optimization problems are abundant in telecommunications (Resende and Pardalos, 2006). Among optimization problems, combinatorial optimization arises in numerous application domains such as IP routing, network design, and network facility location. Much progress has been made in recent years in finding provably optimal solutions to combinatorial optimization problems employing techniques such as branch and bound, branch and cut, cutting planes, and dynamic programming, as well as provably near-optimal solutions using approximation algorithms. Nevertheless, many combinatorial optimization problems arising in telecommunications can benefit from heuristic methods that quickly produce good-quality solutions. Many modern heuristics for combinatorial optimization are based on guidelines provided by metaheuristics.

Metaheuristics (Glover and Kochenberger, 2003; Gendreau and Potvin, 2010) are high-level procedures that coordinate simple heuristics, such as local search, to find solutions that are of better quality than those found by the simple heuristics alone. Many metaheuristics have been introduced in the last thirty years. Among these, we find GRASP, simulated annealing, tabu search, variable neighborhood search, scatter search, path-relinking, iterated local search, ant colony optimization, swarm optimization, and genetic algorithms.

A variant of genetic algorithm, introduced in the last 10 years, is the so-called *biased random-key genetic algorithm* or BRKGA (Gonçalves and Resende, 2010a). BRKGA heuristics encode a solution of the combinatorial optimization problem as a vector of random keys, i.e. a vector with real-valued components randomly generated in the interval $[0, 1)$. They search the solution space of the combinatorial optimization problem indirectly, by exploring the continuous unit hypercube and

Date: November 30, 2010. Revised February 3, 2011.

Key words and phrases. Optimization in telecommunications, genetic algorithm, biased random-key genetic algorithm, random keys, combinatorial optimization, heuristics, metaheuristics.

AT&T Labs Research Technical Report.

mapping continuous solutions to discrete solutions of the combinatorial optimization problem using a simple heuristic called a *decoder*. The decoder is designed to guarantee that, given as input a vector of random keys, it produces a feasible discrete solution of the combinatorial optimization problem. This way there is a clear separation between the problem independent portion of the BRKGA and the problem dependent part, so these methods can be seen as high-level procedures that coordinate simple heuristics to find solutions that are of better quality than those found by the simple heuristics alone.

Random-key genetic algorithms can be traced back to the genetic algorithm with random keys, proposed by Bean (1994), for sequencing problems. Biased random-key genetic algorithms differ from Bean's algorithm in the way parents are chosen for crossover. In a BRKGA one parent is always chosen from a set of elite individuals in the population, while in Bean's algorithm both parents are selected from the entire population. BRKGAs were first introduced by Gonçalves and Beirão (1999) and Gonçalves and Almeida (2002) for sequencing problems and by Buriol et al. (2002; 2005) and Gonçalves and Resende (2004) for problems not involving sequencing.

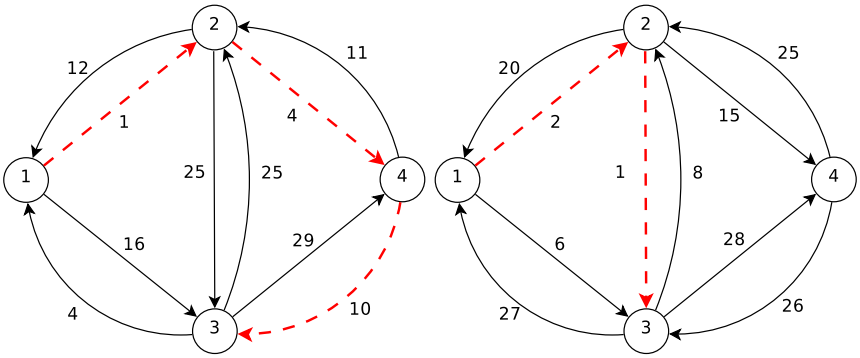
The paper is organized as follows. In Section 2 we describe biased random-key genetic algorithms. This is followed by five sections, each presenting a BRKGA for an optimization problem that arises in telecommunications. Section 3 considers the weight setting problem for routing in IP networks with the Open Shortest Path First (OSPF) routing protocol. Section 4 addresses the design of survivable IP networks where routing is done with the OSPF protocol. In Section 5, we consider the problem of locating content distribution centers in a network. Section 6 describes a BRKGA for the regenerator location problem in optical networks. Finally, Section 7 considers the problem of routing and assigning wavelengths to a set of lightpaths in an optical network. Concluding remarks are made in Section 8.

2. Biased random-key genetic algorithms

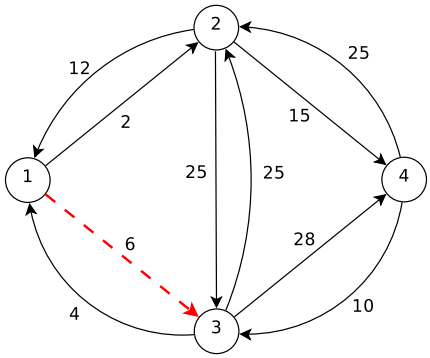
Genetic algorithms with random keys, or *random-key genetic algorithms* (RKGA), were first introduced by Bean (1994) for solving combinatorial optimization problems involving sequencing. In a RKGA, chromosomes are represented as vectors of randomly generated real numbers in the interval $[0, 1)$. A deterministic algorithm, called a *decoder*, takes as input a solution vector and associates with it a feasible solution of the combinatorial optimization problem for which an objective value or *fitness* can be computed. In a minimization (resp. maximization) problem, we say that solutions with smaller (resp. larger) objective function values are more fit than those with larger (resp. smaller) values.

Consider the example in Figure 1 of encoding/decoding for a routing problem on a graph $G = (V, E)$. In this problem links $e \in E$ are assigned weights $\mathcal{W}_e$ and routes are computed by following least-weighted routes. A solution is encoded as a random real-valued $|E|$-vector $\mathcal{X}$ where $\mathcal{X}_e \in [0, 1)$ for all $e \in E$ and are decoded into integer $|E|$-vectors of link weights $\mathcal{W}$ using the decoder $\mathcal{W}_e = \lceil 30\mathcal{X}_e \rceil$, for $e \in E$. The example shows encoding/decoding for two parent solutions and the child that results from the crossover of the two parents. The shortest paths between nodes 1 and 3 in the three solutions are shown by dashed links.

A RKGA evolves a population of random-key vectors over a number of iterations, called *generations*. The initial population is made up of p vectors of random-keys. Each component of the solution vector is generated independently at random in



(a) Parent 1 (b) Parent 2



(c) Child

FIGURE 1. Example of encoding/decoding of parent and child solutions for a graph routing problem. Encoded real-valued random vectors $\mathcal{X}$ are decoded into an integer vector of link weights $\mathcal{W}$ with decoder $\mathcal{W}_i = \lceil 30\mathcal{X}_i \rceil$. Parent 1 has chromosome $\mathcal{X} = \{.031, .370, .503, .116, .832, .827, .113, .338, .950, .326\}$ which is decoded as $\mathcal{W} = \{1, 12, 16, 4, 25, 25, 4, 11, 29, 10\}$. Parent 2 has chromosome $\mathcal{X} = \{.035, .666, .181, .900, .019, .252, .471, .821, .907, .855\}$ which is decoded as $\mathcal{W} = \{2, 20, 6, 27, 1, 8, 15, 25, 28, 26\}$. Child obtained by mating parents 1 and 2 with coin flips $\{THTHHHTTTH\}$ resulting in chromosome $\mathcal{X} = \{.035, .370, .181, .116, .832, .827, .471, .821, .907, .326\}$ which is decoded as $\mathcal{W} = \{2, 12, 6, 4, 25, 25, 15, 25, 28, 10\}$. The shortest paths from node 1 to node 3 for the solutions are indicated by the dashed arrows.

the real interval $[0, 1)$. After the fitness of each individual is computed by the decoder in generation k, the population is partitioned into two groups of individuals (see Figure 2): a small group of p_e *elite* individuals, i.e. those with the best fitness values, and the remaining set of $p - p_e$ *non-elite* individuals. To evolve the

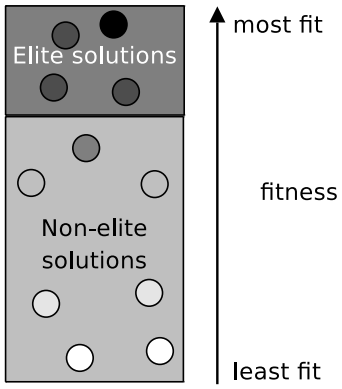


FIGURE 2. Population of p solutions is partitioned into a smaller set of p_e elite (most fit) solutions and a larger set of $p-p_e$ non-elite (least fit) solutions.

population, a new generation of individuals must be produced. All elite individual of the population of generation k are copied without modification to the population of generation $k+1$ (see Figure 3). RKGAs implement mutation by introducing *mutants* into the population. A mutant is simply a vector of random keys generated in the same way that an element of the initial population is generated. At each generation, a small number (p_m) of mutants is introduced into the population (see Figure 3). With the p_e elite individuals and the p_m mutants accounted for in population $k+1$, $p-p_e-p_m$ additional individuals need to be produced to complete the p individuals that make up the new population. This is done by producing $p-p_e-p_m$ offspring through the process of mating or crossover (see Figure 4).

Bean (1994) selects two parents at random from the entire population to implement mating in a RKGA and allows a parent to be selected more than once in a given generation. A *biased random-key genetic algorithm,* or BRKGA (Gonçalves and Resende, 2010a), differs from a RKGA in the way parents are selected for mating. In a BRKGA, each element is generated combining one element selected at random from the elite partition in the current population and one from the non-elite partition. We say the selection is *biased* since one parent is always an elite individual. Repetition in the selection of a mate is allowed and therefore an individual can produce more than one offspring in the same generation. *Parameterized uniform crossover* (Spears and DeJong, 1991) is used to implement mating in BRKGAs. Let $\rho_e > 0.5$ be the probability that an offspring inherits the vector component of its elite parent. Let n denote the number of components in the solution vector of an individual. For $i = 1, \ldots, n$, the i-th component $c(i)$ of the offspring vector c takes on the value of the i-th component $e(i)$ of the elite parent e with probability ρ_e and the value of the i-th component $\bar{e}(i)$ of the non-elite parent $\bar{e}$ with probability $1-\rho_e$.

When the next population is complete, i.e. when it has p individuals, fitness values are computed by the decoder for all of the newly created random-key vectors and the population is partitioned into elite and non-elite individuals to start a new generation. Figure 5 shows a flow diagram of the BRKGA framework with a clear

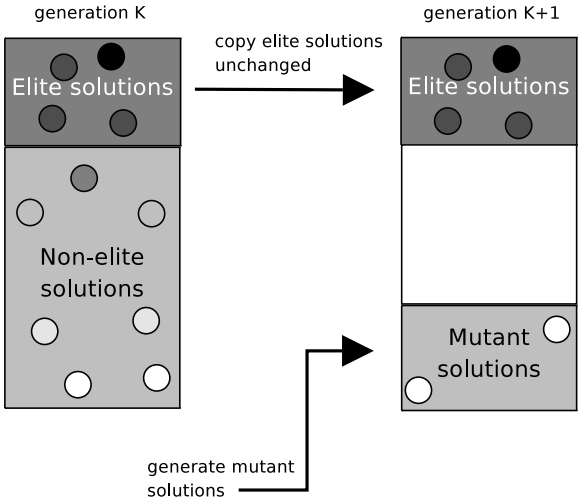


FIGURE 3. All p_e elite solutions from population k are copied unchanged to population $k+1$ and p_m mutant solutions are generated in population $k+1$ as random-key vectors.

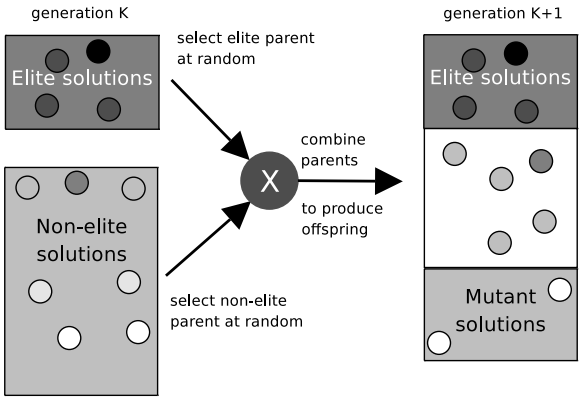


FIGURE 4. To complete population $k+1$, $p - p_e - p_m$ offspring are created by combining a parent selected at random from the elite set of population k with a parent selected at random from the non-elite set of population k. Parents can be selected for mating more than once per generation.

separation between the problem dependent and problem independent components of the method.

A BRKGA searches the solution space of the combinatorial optimization problem indirectly by exploring the continuous n-dimensional hypercube, using the decoder to map solutions in the hypercube to solutions in the solution space of the combinatorial optimization problem where the fitness is evaluated.

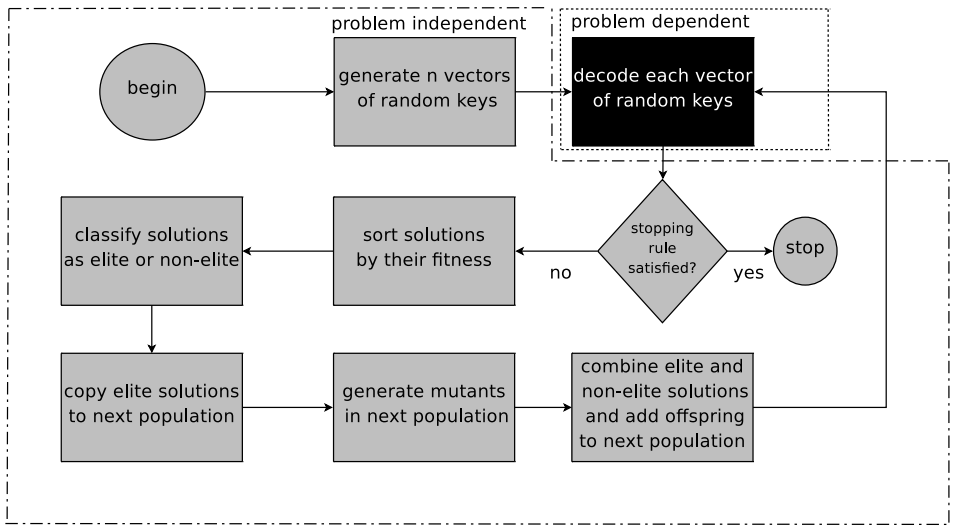


FIGURE 5. Flowchart of biased random-key genetic algorithm with problem independent and problem dependent components.

To describe a BRKGA for a specific combinatorial optimization problem, one needs only to show how solutions are encoded as vectors of random keys and how these vectors are decoded to feasible solutions of the optimization problem. In the next sections, we describe biased random-key genetic algorithms for optimization problems arising in telecommunications.

3. Weight setting problem in IP routing

The Internet is made up of routing domains, called autonomous systems (ASes), which are networks consisting of routers and links connecting the routers. When customer and peer routers are considered, ASes can have thousands of routers and links. ASes interact to control and deliver Internet Protocol (IP) traffic. They typically fall under the administration of a single institution, such a service provider. Neighboring ASes use the Border Gateway Protocol (BGP) to route traffic between them.

The goal of intra-domain traffic engineering is to improve user performance and make more efficient use of network resources within an AS. Interior Gateway Protocols (IGPs), such as OSPF (Open Shortest Path First) and IS-IS (Intermediate System-Intermediate System), are commonly used to select the paths along which traffic is routed within an AS.

These routing protocols direct traffic based on link weights assigned by the network operator. Each router in the AS computes shortest paths, using Dijkstra's algorithm (Dijkstra, 1959), and creates destination tables used to direct each IP packet to the next router on the path to its final destination. OSPF calculates routes as follows. An integer weight ranging from 1 to 65,535 ($= 2^{16} - 1$) is assigned to each link. The weight of a path is the sum of the link weights on the path. OSPF requires that each router compute a graph of shortest paths with itself as the root. This graph gives the least weight routes (including multiple routes in case of ties) to all destinations in the AS. In the case of multiple shortest paths

originating at a router, OSPF is usually implemented so that it will accomplish load balancing by splitting the traffic flow over all shortest paths leaving from each router. A common assumption is to consider that traffic is split evenly between all outgoing links on the shortest paths to the destination IP address. OSPF requires routers to exchange routing information with all the other routers in the AS, a requirement for the computation of the shortest paths.

Given a set of traffic demands between origin-destination pairs, the *OSPF weight setting problem* consists in determining weights to be assigned to the links so as to optimize a cost function, typically associated with a network congestion measure. Figure 6 shows an example of two weight settings and the link loads resulting from routing the demands with the OSPF protocol.

Ericsson et al. (2002) and Buriol et al. (2005) describe BRKGA heuristics for the weight-setting problem in OSPF routing. A related BRKGA is described in Reis et al. (2010), where a different routing protocol, *Distributed Exponentially-Weighted Flow Splitting* (DEFT), is used.

3.1. **Problem definition.** Consider a directed network graph $G = (N, A)$ where N denotes the set of nodes (where routers are located) and A denotes the set of links connecting the routers with a capacity c_a for each $a \in A$, and a demand matrix D that, for each pair $(s,t) \in N \times N$, gives the demand $d_{s,t}$ in traffic flow from node s to node t. The *OSPF weight-setting problem* consists in assigning positive integer weights $w_a \in [1, w_{max}]$ to each arc $a \in A$, such that a measure of routing cost is minimized when the demands are routed according to the rules of the OSPF protocol. The routing cost is a function of the link capacities and the total traffic that traverses each link. In OSPF, traffic between nodes s and t is routed on a shortest-weight path connecting these nodes. The OSPF protocol allows for $w_{max} \leq 65535$.

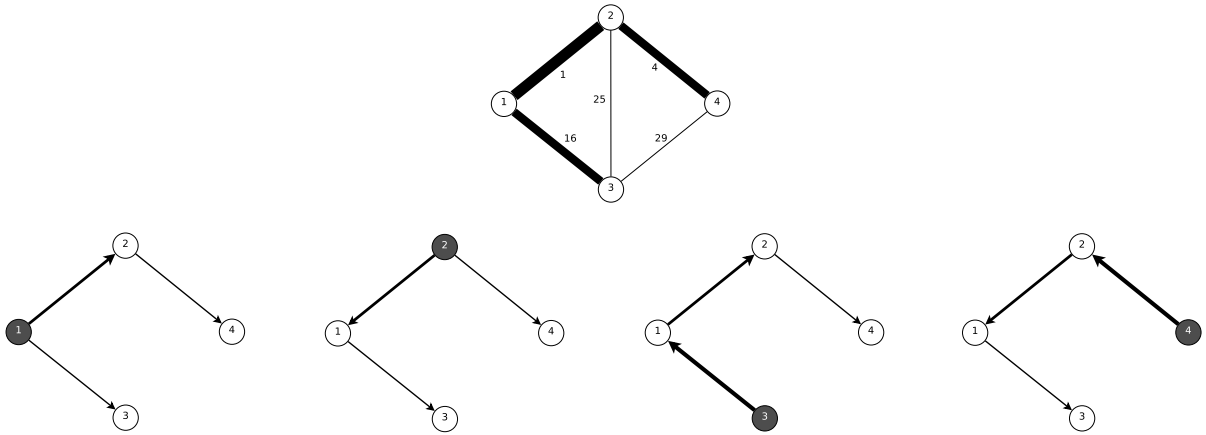
For each pair (s,t) and each arc a, let $f_a^{(st)}$ indicate how much of the traffic flow from s to t goes over arc a. Let l_a be the total load on arc a, i.e. the sum of the flows going over a, and let the trunk utilization rate $u_a = l_a/c_a$. The routing cost in each arc $a \in A$ is taken as the piecewise linear function $\Phi_a(l_a)$, proposed by Fortz and Thorup (2004) and depicted in Figure 7, which increasingly penalizes flows approaching or violating the capacity limits:

$$\Phi_a(l_a) = \begin{cases} u_a, & u_a \in [0, 1/3) \\ 3 \cdot u_a - 2/3, & u_a \in [1/3, 2/3), \\ 10 \cdot u_a - 16/3, & u_a \in [2/3, 9/10), \\ 70 \cdot u_a - 178/3, & u_a \in [9/10, 1), \\ 500 \cdot u_a - 1468/3, & u_a \in [1, 11/10), \\ 5000 \cdot u_a - 16318/3, & u_a \in [11/10, \infty). \end{cases} \tag{1}$$

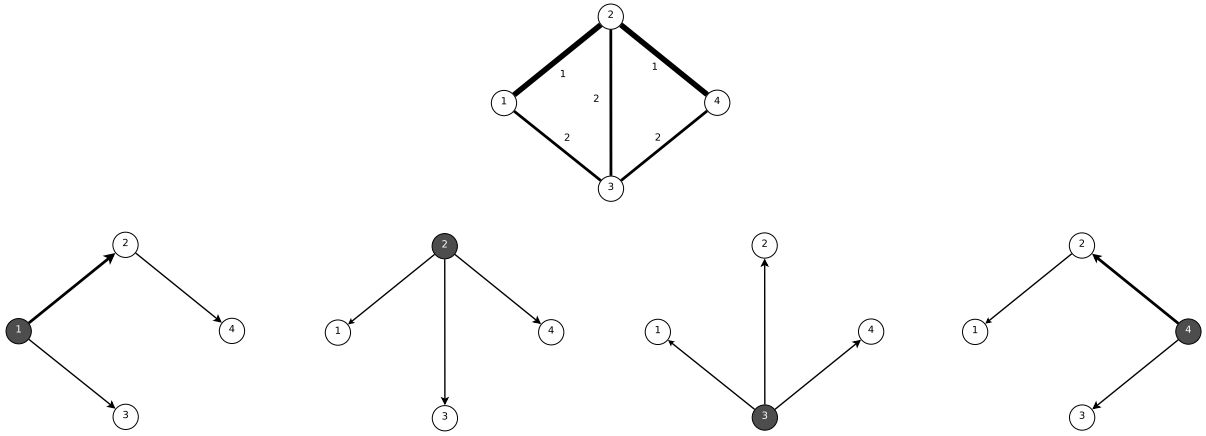
The routing cost is $\Phi = \sum_{a \in A} \Phi_a(l_a)$.

3.2. **Solution encoding.** Each solution is encoded as a vector x of random keys of length $n = |A|$, where the i-th gene corresponds to the i-th link of G.

3.3. **Chromosome decoder.** To decode a link weight w_i from x_i (for $i = 1, \ldots, n$), simply compute $w_i = \lceil x_i \times w_{max} \rceil$. Once link weights are computed, shortest weight (path) graphs from each node to all other nodes in the graph can be derived, traffic can be routed on least weight paths, the total traffic on each link computed, resulting in a routing cost which is the fitness of the solution. Buriol et al. (2005)



(a) Weight setting configuration 1



(b) Weight setting configuration 2

FIGURE 6. Each node must send one unit of flow to each other node in the network. For each configuration, the network on top shows link weights and total flow on each link. Links on the top network are bi-directional and weights are symmetric. The flows are determined by routing the demands on shortest weight paths (see four networks on the bottom). Assuming the links have equal capacities, configuration 2 has a smaller congestion cost than configuration 1. Configuration 2 also has less maximum utilization.

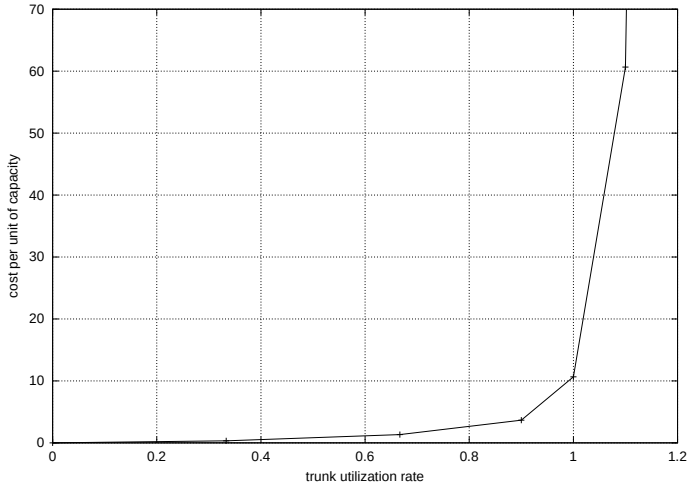


FIGURE 7. Piecewise linear function $\Phi_a(l_a)$.

apply a fast local search to the solution in an attempt to further reduce the routing cost of OSPF routing. Let A^* be the set of five links with the highest routing cost values. For each link $i \in A^*$, a local improvement heuristic attempts to increase w_i by one unit at a time in a specified range and adjust the traffic accordingly. If the total routing cost can be reduced this way, the new weight is accepted, a new set A^* is constructed, and the process repeats itself. If, after scanning the five links, the cost cannot be reduced, then the procedure stops. This fast local search was adapted for DEFT routing in Reis et al. (2010).

3.4. **Experimental results.** Ericsson et al. (2002) compare routing solutions produced by their BRKGA for the 13 test problems proposed by Fortz and Thorup (2004) with lower bounds derived by solving a multicommodity flow linear program (LP), the tabu search heuristic of Fortz and Thorup, and the simple heuristics *UnitOSPF*, *InvCapOSPF*, and *RandomOSPF*. The BRKGA was run for 700 generations on each instance and easily outperformed the simple heuristics, finding solutions comparable with those of Fortz and Thorup. These solutions were close to the LP lower bounds for a wide range of traffic demands. By running BRKGA independently 9 times for 8000 generations on each one of the instances, the BRKGA was shown to produce better solutions than Fortz and Thorup on all 9 runs. Figure 8 shows the fitness of the best solution of each of the nine runs on an instance proposed by Fortz and Thorup (2004) as a function of CPU time. The best solution found was closer to the LP lower bound than to the solution produced by the search heuristic of Fortz and Thorup.

Buriol et al. (2005) test their BRKGA on the same 13 test instances considered by Fortz and Thorup (2004) and Ericsson et al. (2002). Figure 9 shows that the new decoder with the fast local search finds better solutions than the BRKGA of Ericsson et al. Furthermore, Buriol et al. (2005) show that given a target solution value, the new BRKGA is also faster than the BRKGA of Ericsson et al. Finally, they show results of experiments comparing run-time distributions for the BRKGA and the tabu search of Fortz and Thorup. Using three target values on a large real

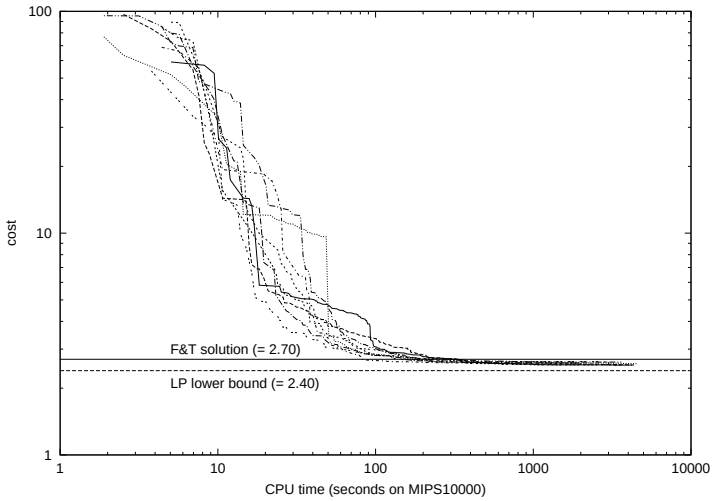


FIGURE 8. Fitness values of best solution on nine independent runs of the BRKGA of Ericsson et al. (2002) on a test instance proposed in Fortz and Thorup (2004). The figure shows the best value obtained by the local search of Fortz and Thorup (2004) as well as the value of linear programming lower bound.

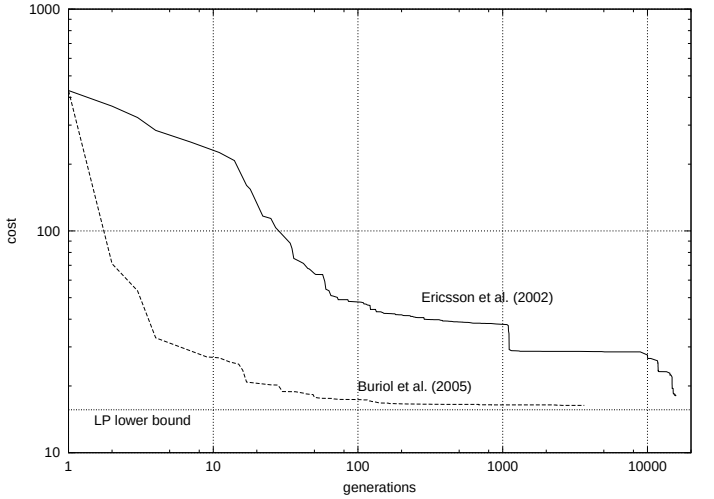


FIGURE 9. Cost of best solution found by the BRKGA of Ericsson et al. (2002) and the the BRKGA of Buriol et al. (2005) as a function of number of iterations for a one hour run on a 196-MHz MIPS R10000 processor.

instance, the experiments show that the tabu search distribution has a long tail while the distribution for the BRKGA does not.

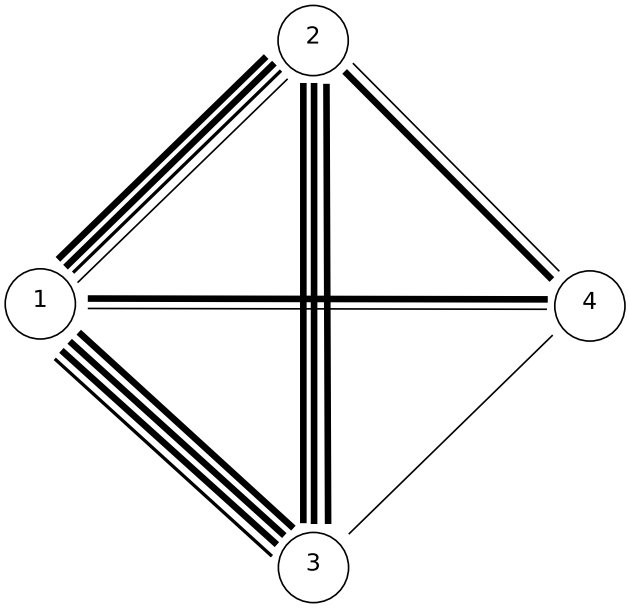


FIGURE 10. A four-node network with three link-types (denoted with three different line widths) that is survivable to single arc failures. An arc failure knocks out all link types on that arc and each remaining working arc has to have enough installed capacity to handle the additional traffic that will be pass through it to avoid the failed arc.

Reis et al. (2010) compare their BRKGA for DEFT routing with the BRKGA of Buriol et al. (2005) for OSPF routing. They show results for the 13 test problems used by previous papers and confirm that DEFT routing can achieve solutions that result in less congestion than OSPF routing.

4. Design of survivable IP networks

Given a set of nodes in a network, a traffic matrix estimating the demand, or traffic, between pairs of these nodes, a set of arcs, each having endpoints at a pair of the given nodes, a set of possible fiber link types, each with an associated capacity and cost per unit of length, and a set of failure configurations, the survivable network design problem seeks to determine how many units of each cable type will be installed in each link such that all of the demand can be routed on the network under the no failure and all failure modes such that the total cost of the installed fiber is minimized. Figure 10 illustrates a four node survivable network with three link types. Buriol et al. (2007) proposed a BRKGA to design survivable networks where traffic is routed using the Open Shortest Path First (OSPF) protocol and there is only one link type. Andrade et al. (2006) extended this BRKGA to handle composite links, i.e. the case where there are several fiber types. Four decoders are proposed by Andrade et al.

4.1. **Problem definition.** Given a directed graph $G = (V, E)$, where V is the set of routers and E is the set of potential arcs where fiber can be installed, and a demand

matrix D, that for each pair $(u, v) \in V \times V$, specifies the demand $D_{u,v}$ between u and v. Arc $e \in E$ has length d_e. Link types are numbered $1, \ldots, T$. Link type i has capacity c_i and cost per unit of length p_i. We make two assumptions about the capacities and costs per unit length. First, given capacities $c_1 < c_2 < \ldots < c_T$ and prices $p_1 < p_2 < \ldots < p_T$, we assume that $(p_T/c_T) < (p_{T-1}/c_{T-1}) < \ldots < (p_1/c_1)$, i.e., the price per unit of capacity is smaller for links with greater capacities. Second, given capacities $c_1 < c_2 < \ldots < c_T$, we assume that $c_i = \alpha c_{i-1}$ ($\alpha \in \mathbb{N}, \alpha > 1$), i.e., the capacities are multiples of each other by powers of the integer α.

We wish to determine integer OSPF weights $w_e \in [1, 65535]$ as well as the number of copies of each link type to be deployed at each arc such that when traffic is routed according to the OSPF protocol in a no-failure or any single failure situation there is enough installed capacity to move all of the demand. Furthermore, we want the total cost of the installed capacity to be minimum.

4.2. **Solution encoding.** Assume arcs in E are numbered $1, \ldots, |E|$. A solution of the survivable network design problem is encoded as a vector x of $|E|$ random keys. The i-th key corresponds to the i-th arc.

4.3. **Chromosome decoder.** To produce the OSPF weight w_i of the i-th arc, scale the random key by the maximum weight, i.e. set $w_i = \lceil x_i \times 65535 \rceil$. For the no-failure mode and each failure mode, route the traffic using the OSPF protocol using the computed arc weights, compute the loads on each arc and record the maximum load over the no-failure and all failure modes. For each arc, determine an optimal allocation of link types such that the resulting capacity of the set of composite links is enough to accommodate the maximum load on the arc. To do this on a particular arc, start with load l. Use as much as possible of the highest capacity link type without exceeding the load. This means that $\lfloor l/c_T \rfloor$ units of link t are used. Compute the cost if we were now to satisfy the load using the current link type and save this configuration and its cost but do not deploy this link type for now. Update the remaining load ($l := l - \lfloor l/c_T \rfloor$), and repeat the operation on the next link type ($T - 1$), until the link type (1) with smallest capacity is reached. Then, satisfy the remaining load with $\lceil l/c_1 \rceil$ units of link type 1. Of the several configurations considered for the arc, deploy the one with smallest cost configuration. Finally, when all arcs have link deployments, add up the costs of these deployments and return the sum as the fitness of the solution.

4.4. **Experimental results.** Since this was the first heuristic proposed in the literature for this problem, Buriol et al. (2007) compare network designs produced with their BRKGA with those produced by a similar process where instead of finding good OSPF weights with the BRKGA, link weights are set in one case to unit (*UNIT*) and randomly (*RAND*) in another. They also compare their solutions with a simple lower bound (*LB*). Four networks of sizes varying from 10 nodes and 90 links to 71 nodes and 350 links make up the benchmark test set. For each network, four instances were created: one with no failures, one with both single router and single link failures, one with single link failures and no router failure, and one with single router failures and no link failure. The results show that the solutions produced by the BRKGA are superior to those produced with the other heuristics. For example, a 1000-generation run with a 500-element population produced for one of the instances with no failure the following ratios of solution values: 1.64 for *RAND*:BRKGA, 1.82 for *RAND*:BRKGA, and 1.94 for BRKGA:*LB*. Figure 11

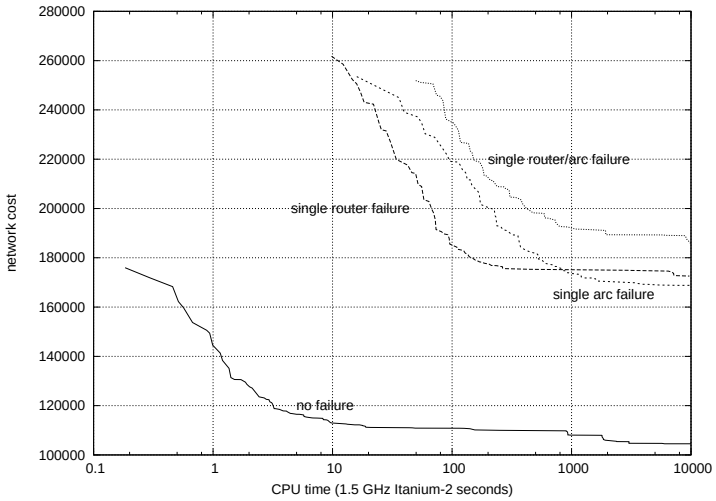


FIGURE 11. Network cost of BRKGA as a function of CPU time for unprotected network and three networks having different types of protection: single router failure, single link failure, and single router or link failure.

illustrates the additional cost of adding different types of protection on a 74 node, 278 edge network with 18 routers that make up 306 router pairs with traffic requirements (Buriol et al., 2007). Networks were designed by running the BRKGA variants for 10,000 seconds on a 1.5 GHz Itanium-2 processor. To add protection against single link failures costs 61% more than having no protection. Protection against single router failure costs 65% more and against both single router or single link failure is 78% more expensive than having no protection at all.

Andrade et al. (2006) show the results of an experiment on a real network with 54 routers and 278 arcs. Three link types were considered. All four decoders were tested and the so-called *min cost* decoder achieved the best results among the decoders tested.

5. Redundant server location for content distribution

Breslau et al. (2011) study two new facility location problems that arise in telecommunications. A BRKGA is proposed for these problems. In these applications a customer must be served by a pair of facilities. In addition, the service routes from the facilities to the customer must be vertex-disjoint. In this context, we wish to find a minimum-size set of facilities such that each customer has associated with it at least two facilities where the service routes from the facilities to the customer are vertex-disjoint.

One of the applications discussed in Breslau et al. (2011) is redundant server location for content distribution. Suppose we wish to operate a robust system to distribute real-time content, such as television broadcasts, on a network that does not have a fast scheme to recover from link or node failures. A common characteristic of robustness is survivability to single link or node failures. To achieve this we place multiple copies of our content in the network. Since content-hosting

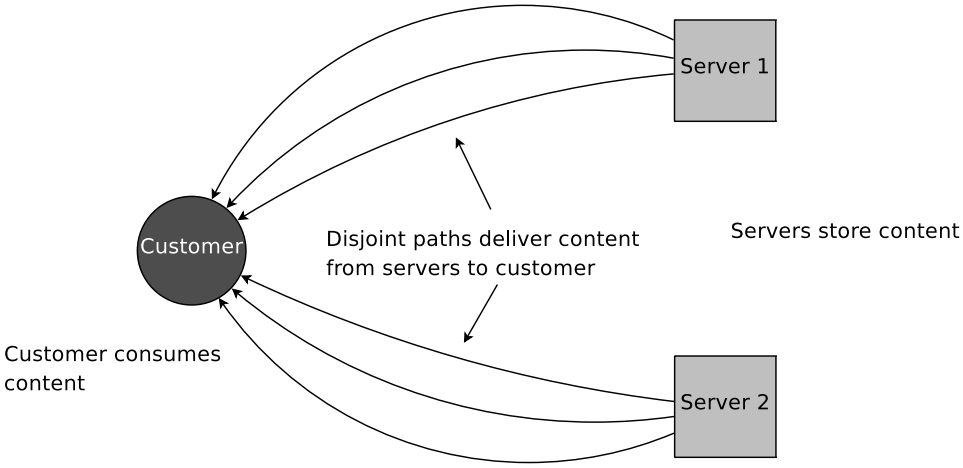


FIGURE 12. Servers store digital content that is consumed by customers. Each customer requires that two servers be assigned to deliver content to it. These servers must be such that all paths from the first server to the customer are disjoint from all paths from the second server to the customer. That way a single edge failure will not affect service to the customer.

facilities are expensive, we wish to deploy as few of them as possible, as long as the required resilience is achieved. A customer c is said to be covered if there exist at least two content-hosting facilities f_1 and f_2 such that the paths from f_1 to f_2 are such that they only share node c. We wish to cover all customers with as few facilities as possible.

5.1. **Problem definition.** Suppose we are given a network modeled as an undirected graph $G = (V, A)$ and two sets C and F such that $C, F \subseteq V$. C is the set of customer locations and F the set of potential facility locations. Furthermore, suppose that for each pair (c, f) such that $c \in C$ and $f \in F$ we are given a set $P(c, f)$ of simple directed paths from f to c in G. These paths could, for example, be the set of all OSPF shortest-path routes from c to f. We say (f_1, f_2) covers c in a pathwise-disjoint way if there exist paths $p_1 \in P(f_1, c)$ and $p_2 \in P(f_2, c)$ that have no common vertex except c. The pair (f_1, f_2) covers c in a setwise-disjoint way if all paths $p_1 \in P(f_1, c)$ and $p_2 \in P(f_2, c)$ share only vertex c. A subset $F' \subseteq F$ is called a pathwise-disjoint cover for C if for every $c \in C$ there exists a pair (f_1, f_2), where $f_1, f_2 \in F'$ such that (f_1, f_2) covers c in a pathwise-disjoint way. Similarly, a subset $F' \subseteq F$ is called a setwise-disjoint cover for C if for every $c \in C$ there exists a pair (f_1, f_2), where $f_1, f_2 \in F'$ such that (f_1, f_2) covers c in a setwise-disjoint way. In the *pathwise-disjoint facility location problem*, we are given G, C, F, and the sets $P(c, f)$ and want to find a pathwise-disjoint cover of minimum size for C, if such a cover exists. Similarly, in the *setwise-disjoint facility location problem*, we are given G, C, F, and the sets $P(c, f)$ and want to find a setwise-disjoint cover of minimum size for C, if such a cover exists. These facility location problems are not only NP-hard but also strongly inapproximable in the worst-case (Breslau et al., 2011).

5.2. **Solution encoding.** Assume the potential facility locations in F are numbered $1, \ldots, |F|$. A solution of the pathwise (setwise) facility location problem is encoded as a vector x of $|F|$ random keys. The i-th key corresponds to the i-th potential facility location.

5.3. **Chromosome decoder.** The decoder takes as input a vector x of $|F|$ random keys and returns a subset $F' \subseteq F$ of facilities that is a cover for C.

A greedy algorithm for this problem is proposed in Breslau et al. (2011). It initializes the set $F' \subseteq F$ to the empty set. Then, as long as F' does not cover all customers $c \in C$, find a facility $f \in F \setminus F'$ such that $F' \cup \{f\}$ covers a maximum number of yet-uncovered customers. Ties are broken by facility index. If the number of yet-uncovered customers that become covered is at least one, add f to F'. Otherwise, find a pair of facilities $(f_1, f_2) \in F \setminus F'$ such that $F' \cup \{f_1\} \cup \{f_2\}$ covers a maximum number of yet-uncovered customers. Ties are broken first by the index of f_1, then by the index of f_2. If such a pair does not exist, then the problem is infeasible. Otherwise, add f_1 and f_2 to F'.

The decoder initializes set F' according to the values of the random keys. For $f = 1, \ldots, |F|$, if key $x_f \geq 0.5$ then f is added to set F'. If the resulting set F' is a cover for C, then the decoder returns it and stops. Otherwise, the above greedy algorithm is applied starting with the facilities in F' instead of the empty set.

5.4. **Experimental results.** Breslau et al. (2011) tested the BRKGA for disjoint-path facility location on synthetic and real-world instances modeling two different types of data networks and having distinct characteristics. Synthetic networks had up to 558 nodes while the largest real-world network had about 1000 nodes.

The BRKGA was compared with a greedy algorithm, and a more sophisticated heuristic, called the *double-hitting set heuristic*, with respect to accuracy, execution time, and cost reduction with respect to a trivial solution.

With respect to accuracy, on 41 problem instance classes, the BRKGA was on average never over 2.5% above a strong lower bound (compared to 3.7% for the double-hitting set heuristic and 2.8% for greedy). Of the 560 synthetic instances, the BRKGA found the best solution on 527 (versus 491 for the double-hitting set and 499 for greedy). On the real-world instances, the results were similar to those for the synthetic instances, i.e. the BRKGA was slightly better than the double-hitting set heuristic in many comparison (with the exception of the largest instances, too large for the single-thread implementation of BRKGA to handle). Both the BRKGA and the double-hitting set heuristic outperformed the greedy algorithm on the real-world instances.

With respect to execution times, the BRKGA was slowest of the three proposed heuristics, taking over a day on a 531 node instance. In fact, it was not even run on the largest real-world instances. This, of course, can be mitigated with a multi-thread implementation of the BRKGA. Our experience with another set covering problem (Resende et al., 2010) showed a speedup of about 40 using 120 processors. Since the publication of Breslau et al. (2011), we have developed a new multi-threaded BRKGA implementation which can solve the largest 1000-node instance to optimality (proven by a tight lower bound) in about 1 hour and 43 minutes on a single 2.27GHz Xeon X7560 processor and 4 minutes 25 seconds with 64 processors (a factor of 23 speedup).

Perhaps most importantly, compared to the trivial solution of locating a facility in each node of the network, the BRKGA was able to achieve significant reductions. On the synthetic instances, the smallest savings amounted to 33% while the largest was 76%. If the set-disjoint variant of the problem is considered, the savings are even slightly better. These savings were even more pronounced on the real-world instances, were the BRKGA found solution that had savings of 85-90% with respect to the trivial solution.

6. Regenerator location

Telecommunication systems transmit information with optical signals. Because of attenuation, the strength of a signal deteriorates, losing power, the further it gets from the source. To enable the signal to arrive at its intended destination with sufficient strength, it may be necessary to regenerate the signal one or more times using regenerators. Since regenerators are relatively expensive, it is desirable to deploy as few of them as possible in the network. In the *regenerator location problem* (RLP) we seek paths connecting all pairs of nodes in the network with a minimum number of regenerators. Duarte et al. (2010) proposed a BRKGA for the RLP.

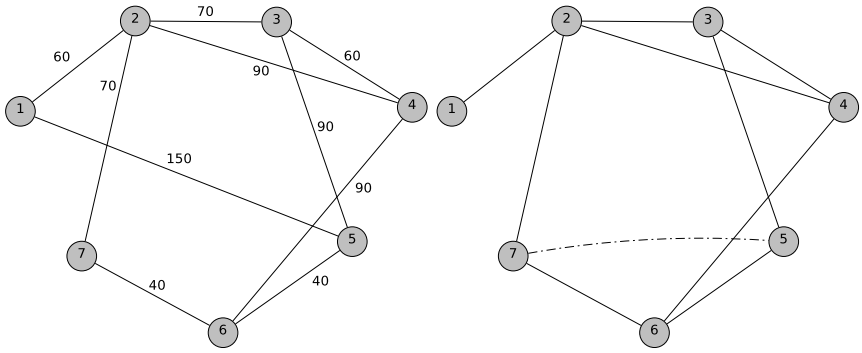
6.1. **Problem definition.** Consider an undirected graph $G = (V, E)$, where V is the node set, E is the set of edges. Each edge $(i, j) \in E$ has a real-valued length $d_{i,j} \in \mathbb{R}^+$. A parameter $D \in \mathbb{R}^+$ specifies the maximum length that a signal can travel before its quality deteriorates such that regeneration is required. The *regenerator location problem* (RLP) consists in determining paths that connect all pairs of nodes in the graph and, if necessary, locating single regenerators at some of those nodes. The regenerator location problem was shown to be NP-hard by Flammini et al. (2009) and Chen et al. (2010).

Between each pair of nodes $\{s, t\} \in V \times V$, a *path* $\{(s, v_1), (v_1, v_2), \ldots, (v_k, t)\}$ connecting these nodes is formed by one or more path segments. A *path segment* consists of a sequence of consecutive edges $\{(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), \ldots, (v_{q-1}, v_q)\}$ in the path, satisfying the condition that the *total length* of the segment

$$d_{v_i,v_{i+1}} + d_{v_{i+1},v_{i+2}} + \cdots + d_{v_{q-1},v_q} \leq D.$$

If the total length of the path is not greater than D, then the path consists of a single path segment. Otherwise, it consists of two or more path segments and one or more regenerators will be located in the internal nodes of the path, separating consecutive path segments.

Figure 13(a) shows an example of a network with seven nodes and a maximum distance parameter $D = 100$. The numbers beside the edges represent their lengths. Note that the length $d_{1,5} = 150$ of edge $(1, 5)$ is greater than D and therefore it cannot be part of any path. In the figure we can see that the shortest path from node 1 to node 3 is $\{(1, 2), (2, 3)\}$ with a total length of $60 + 70 = 130 > 100$. Therefore, it must be decomposed into two path segments, $\{(1, 2)\}$ and $\{(2, 3)\}$, and a regenerator must be placed in node 2 so that nodes 1 and 3 can be connected using this path. Since edge $(1, 5)$ cannot be part of any path, then to connect nodes 1 and 5 we observe that the shortest feasible path is $\{(1, 2), (2, 3), (3, 5)\}$ with total length $60 + 70 + 90 = 220$. Since $60 + 70 > 100$ and $70 + 90 > 100$, this path must be decomposed into three path segments using two regenerators, one at node 2 and one at node 3. On the other hand, we can connect nodes 5



(a) Graph $G = (V, E)$ (b) Graph $M = (V, E')$

FIGURE 13. Example of a seven node graph G with edges lengths and corresponding communication graph M.

and 7 without needing a regenerator using path $\{(5, 6), (6, 7)\}$ which has a length of $40 + 40 = 80$. Finally, note that placing regenerators in nodes 2 and 7 allows for communication between all pairs of nodes in the graph.

Chen et al. (2010) introduce the *communication graph* to describe algorithms for the RLP. To generate the unweighted communication graph $M = (V, E')$ corresponding to the input graph $G = (V, E)$, first delete all edges that have length greater than D. Add an edge between each non-adjacent pairs of nodes of length equal to the length of the corresponding shortest path if the path length in G is not greater than D. Finally, disregard all length information from M. If the resulting communication graph M is complete, then there is no need for any regenerator. If the resulting graph is not connected, then the problem is infeasible. Alternatively, if the resulting graph is connected but not complete, then one or more regenerators will be required.

Figure 13(b) represents the communication graph M that results from the graph in the example of Figure 13(a). The dashed link in the graph between nodes 5 and 7 is the only link added during the procedure to create M from G. Since M is not complete, then one or more regenerators are needed to enable communication between all node pairs in G.

6.2. **Solution encoding.** Assume nodes in V are numbered $1, \ldots, n$, where $n = |V|$. A solution of the regenerator location problem is encoded as a vector x of n random keys. The i-th key corresponds to the i-th node.

6.3. **Chromosome decoder.** Chen et al. (2010) propose a greedy algorithm for the RLP. The decoder described below makes use of this greedy algorithm. The greedy algorithm takes as input the set of pairs of nodes that are not directly connected (NDC) in the communication graph and builds a set R of regenerator nodes, one node at a time. At each iteration, the procedure determines a node u^* whose inclusion in R enables the connection of the largest number of yet unconnected pairs in the communication graph. Node u^* is added to R and the communication graph is updated by adding to it the edges connecting those yet unconnected pairs that become connected by placing a regenerator in u^*.

```
procedure Decoder
  Data: Communication graph M = (V, E') and vector x of
        random keys
  Result: Set of regenerator nodes: R ⊆ V
1 R ← ∅; C ← V;
2 Ē' = {(i, j) ∈ V × V : (i, j) ∉ E'};
3 Order the nodes in C w.r.t. their associated x value;
4 while Ē' ≠ ∅ do
5 |   Select the next node k ∈ C following the x-order;
6 |   X(k) ← ∅;
7 |   A ← N(k) = {v ∈ V : (k, v) ∈ E'};
8 |   for v ∈ A ∩ R do
9 |   |   A ← A ∪ N(v);
10|   end
11|   for (i, j) ∈ A × A do
12|   |   if (i, j) ∉ E' then
13|   |   |   X(k) ← X(k) ∪ {(i, j)};
14|   |   end
15|   end
16|   R ← R ∪ {k};
17|   C ← C \ {k};
18|   E' ← E' ∪ X(k);
19|   Ē' ← Ē' \ X(k);
20 end
21 return R;
```

Algorithm 1: Pseudo-code for DECODER.

In the decoder, instead of selecting the next node to add to set R as the one which enables the connection of the largest number of yet unconnected pairs in the communication graph, the next node is determined according to the vector of random-keys.

Pseudo-code for the decoder is shown in Algorithm 1. In the initialization of the algorithm the set C consists of all the nodes in V. The elements of C are ordered according to the vector of random keys x. This way, nodes associated with components in x with a relatively large value (i.e., those close to 1) come first. The main loop of the `Decoder` goes from line 4 to 20. It is repeated until the communication graph M is complete, i.e. while $\bar{E}' \neq \emptyset$. In this loop, nodes are selected following the order induced by x. For each candidate node $k \in C$, lines 6 to 15 compute the set $\mathcal{X}(k)$ of yet unconnected pairs in M that would become connected if node k where to house a regenerator. In lines 16 to 19, sets R, C, E', and $\bar{E}'$ are updated to reflect the inclusion of node k in set R. In line 21, the set R of regenerator nodes is returned.

6.4. **Experimental results.** Duarte et al. (2010) compared their implementation of the BRKGA for the RLP with `H1+LS`, the best algorithm in Chen et al. (2010),

and with a GRASP heuristic proposed in Duarte et al. (2010). On the instances tested the GRASP heuristic found the best solutions on all 280 instances whereas `H1+LS` did so on 77% of them. The BRKGA, while taking the longest among the heuristics tested, found the best solution on 89% of the 280 instances.

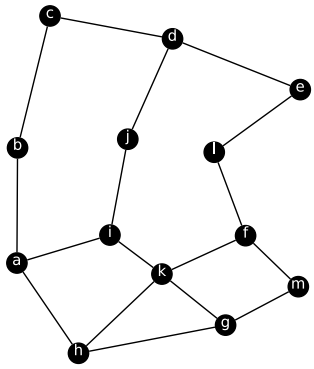
7. Routing and wavelength assignment in optical networks

The problem of routing and wavelength assignment (RWA) in wavelength division multiplexing (WDM) optical networks consists in routing a set of lightpaths (a *lightpath* is an all-optical point-to-point connection between two nodes) and assigning a wavelength to each of them, such that lightpaths whose routes share a common fiber are assigned different wavelengths. Noronha et al. (2010) propose a BRKGA for routing and wavelength assignment with the goal of minimizing the number of different wavelengths used in the assignment (this variant of the RWA is called *min-RWA*). This BRKGA extends the best heuristic in the literature (Skorin-Kapov, 2007) by embedding it into an evolutionary framework.

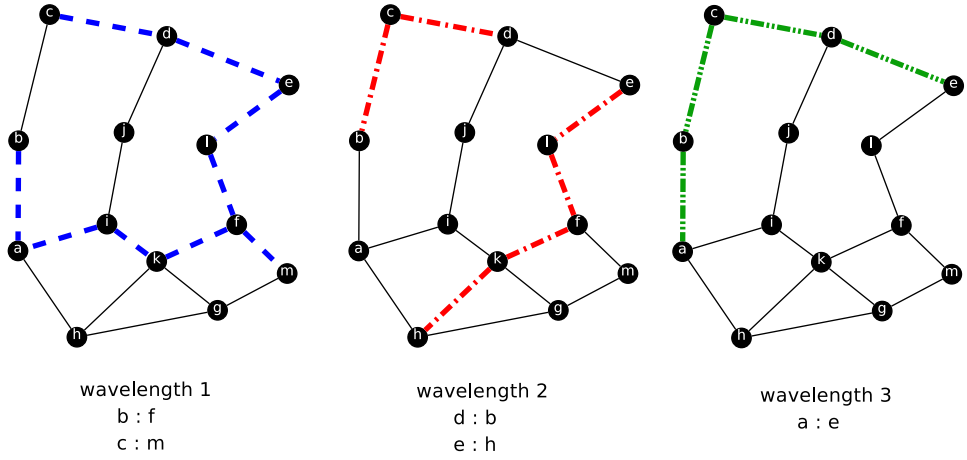
7.1. **Problem definition.** We are given a bi-directed graph $G = (V, E)$ that represents the physical topology of the optical network, where V is the set of nodes and E is the set of fiber links, and a set T of lightpaths to be established. Each lightpath is characterized by its pair of endpoints $\{s, t\} \in V \times V,\ s \neq t$. Each lightpath is routed on a single path from s to t and is assigned the same wavelength for the entire path. If two lightpaths share an arc, they must be assigned different wavelengths. The objective is to minimize the number of wavelengths used.

7.2. **Solution encoding.** A solution of the routing and wavelength assignment problem is encoded in a vector x of $|\mathcal{T}|$ random keys, where $|\mathcal{T}|$ is the number of lightpaths. The key x_i corresponds to the i-th lightpath, for $i = 1, \ldots, |\mathcal{T}|$.

7.3. **Chromosome decoder.** Skorin-Kapov (2007) proposed the current state-of-the-art heuristic for min-RWA. Each wavelength is associated with a different copy of the graph G. Lightpaths that are routed on arc disjoint paths on the same copy of G are assigned the same wavelength. Copies of G are associated with the bins and lightpaths with the items of an instance of the bin packing problem. Therefore, min-RWA can be reformulated as the problem of packing all the lightpath requests in a minimum number of bins. Let $minlength(i)$ be the number of hops in the path with the smallest number of arcs between the endnodes of lightpath i in G. These values are only used for sorting the lightpaths in the decoding heuristics, even though the lightpaths are not necessarily routed on shortest paths. This occurs because whenever a lightpath is routed on a copy of G (or, equivalently, placed in the corresponding bin), all arcs in its route are deleted from this copy to avoid that other lightpaths use them. Therefore, the next lightpaths routed in this copy of G might be routed on a path that is not a shortest path in the original graph G. The classical best fit decreasing heuristic is used to pack the lightpaths. Since the number of lightpaths is usually much greater than the diameter of the graph, there are many lightpaths with the same *minlength* value. In the case of ties, Skorin-Kapov (2007) recommended breaking them randomly. The BRKGA uses the vector of random keys to randomly perturb the values of $minlength(i)$ and get rid of the ties. These values are adjusted as $minlength(i) \leftarrow minlength(i) + x(i)$.



(a) Network and connection requests



(b) Routings and wavelength assignments

FIGURE 14. Subfigure (a) shows the network and the five requested lightpath connections. Three wavelengths are needed to route these lightpaths. Subfigure (b) show wavelength assignments and routes for each lightpath.

7.4. **Experimental results.** Noronha et al. (2010) test their BRKGA extensively on a set of hard instances of the RWA problem. The BRKGA is compared with a multi-start variant *MS-RWA* of the heuristic *BFD-RWA* of Skorin-Kapov (2007) as well as the tabu search based heuristic *2-EDR+TS-PCP* of Noronha and Ribeiro (2006). Noronha et al. observe in their computational experiments that the multi-start heuristic *MS-RWA* was able to improve the results of *BFD-RWA* and also that their BRKGA identifies the relationships between keys and good solutions, converging to better solutions, on average, in 23% less time than *MS-RWA*. The average solution gap observed with the BRKGA was almost 50% of that presented by *2-EDR+TS-PCP*. The experiments also illustrated the robustness of the BRKGA,

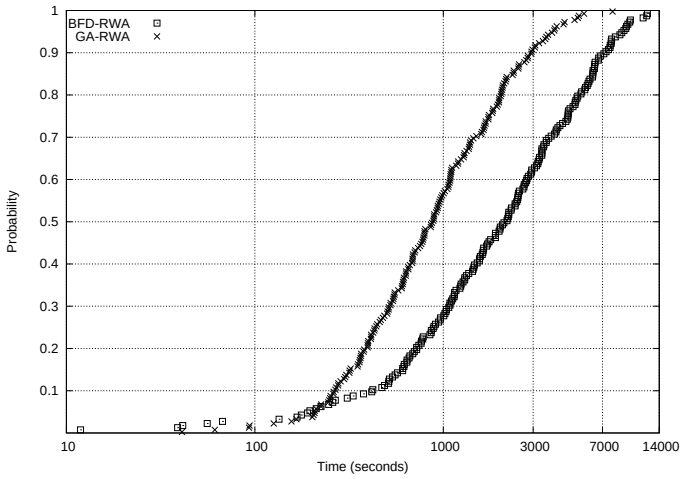


FIGURE 15. Runtime distributions (time-to-target plots) of the BRKGA of Noronha et al. (2010) and the multi-start variant of the best fit decreasing (BFD) heuristic of Skorin-Kapov (2007).

since all versions of the BRKGA (using different parameter settings) obtained good and similar results. Figure 15 shows runtime distributions (time-to-target plots) of the BRKGA of Noronha et al. (2010) and the multi-start variant of the best fit decreasing (BFD) heuristic of Skorin-Kapov (2007). The instance used to produce the runtime distributions is a 100-node network with probability of existing an edge between two nodes equal to 0.05 and probability of existing a lightpath request between two nodes equal to 0.8. Each heuristic was run independently 200 times and was stopped when a solution having cost at least as good as a target value was found. The target solution value used equals 1.093 times the value of the LP lower bound suggested in Bannerjee and Mukherjee (1995). The figure clearly shows that the probability that the BRKGA finds the target solution by a given CPU time is greater than the probability of the multi-start BFD heuristic, from which the BRKGA was derived, doing so.

8. Concluding remarks

The capacity to find optimal or near-optimal solutions to combinatorial optimization problems that arise in telecommunications has far reaching consequences. Given the high-cost of designing and operating telecommunication systems, even a small 1% decrease in costs can result in substantial savings to the system operator. This can potentially translate to better service and lower tariffs to customers. In this paper we survey some recent applications of biased random-key genetic algorithms to find optimal or near-optimal solutions to combinatorial optimization problems that arise in the design and operation of telecommunication systems.

Biased random-key genetic algorithms encode solutions as vectors of random-keys, i.e. randomly generated vectors where each component is a real number in the interval $[0, 1)$. They are characterized by a clear division between a problem independent module and a problem dependent module. The problem dependent module is called a decoder and its role is to map solutions in the continuous unit

hypercube to feasible solutions of the combinatorial optimization problem. The algorithm searches the discrete solution space of the combinatorial optimization problem indirectly by searching the continuous unit hypercube and mapping those solutions to the discrete solution space with the decoder.

To define a biased random-key genetic algorithm, we only need to specify how solutions are encoded and how they are decoded. We show how this is done for five combinatorial optimization problems that arise in telecommunications: the weight setting problem in OSPF routing in IP networks; design of survivable IP networks where routing is done with OSPF; location of redundant servers for content distribution; location of signal regenerators in optical networks; and routing and wavelength assignment in optical networks.

For each problem we show how a biased random-key genetic algorithm can result in an effective heuristic, often finding better solutions than existing heuristics, other times finding same-quality solutions in less CPU time.

In addition to these applications in telecommunications, biased random-key genetic algorithms have been applied to a number of different combinatorial optimization problems, including tollbooth location and tariff assignment in transportation systems (Buriol et al., 2010), scheduling (Gonçalves et al., 2005; Valente et al., 2006; Gonçalves et al., 2008; Valente and Gonçalves, 2008; Mendes et al., 2009; Gonçalves et al., 2010), manufacturing cell formation (Gonçalves and Resende, 2004), packing (Gonçalves, 2007; Gonçalves and Resende, 2009; 2010b), set covering (Resende et al., 2011), and concave network optimization (Fontes and Gonçalves, 2007).

References

D. V. Andrade, L. S. Buriol, M. G. C. Resende, and M. Thorup. Survivable composite-link IP network design with OSPF routing. In *Proceedings of The Eighth INFORMS Telecommunications Conference*, 2006.

D. Bannerjee and B. Mukherjee. Practical approach for routing and wavelength assignment in large wavelength routed optical networks. *IEEE Journal on Selected Areas in Communications*, 14:903–908, 1995.

J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. on Computing*, 6:154–160, 1994.

L. Breslau, I. Diakonikolas, N. Duffield, Y. Gu, M. Hajiaghayi, D.S. Johnson, M.G.C. Resende, and S.Sen. Disjoint-path facility location: Theory and practice. In *ALENEX 2011: Workshop on algorithm engineering and experiments*, January 2011.

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*, 46: 36–56, 2005.

L. S. Buriol, M. G. C. Resende, and M. Thorup. Survivable IP network design with OSPF routing. *Networks*, 49:51–64, 2007.

L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A memetic algorithm for OSPF routing. In *Proceedings of the 6th INFORMS Telecom*, pages 187–188. Citeseer, 2002.

L.S. Buriol, M.J. Hirsch, T. Querido, P.M. Pardalos, M.G.C. Resende, and M. Ritt. A biased random-key genetic algorithm for road congestion minimization. *Optimization Letters*, 4:619–633, 2010.

Si Chen, I. Ljubić, and S. Raghavan. The regenerator location problem. *Networks*, 55:205–220, 2010.

E. Dijkstra. A note on two problems in connection of graphs. *Numerical Mathematics*, 1:269–271, 1959.

A. Duarte, R. Martí, M.G.C. Resende, and R.M.A. Silva. Randomized heuristics for the regenerator location problem. Technical report, AT&T Labs Research, Florham Park, New Jersey, 2010. URL `http://www.research.att.com/~mgcr/doc/gpr-regenloc.pdf`.

M. Ericsson, M. G .C. Resende, and P. M. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *J. of Combinatorial Optimization*, 6: 299–333, 2002.

M. Flammini, A. Machetti, G. Monaco, L. Moscardelli, and S. Zaks. On the complexity of the regenerator placement problem in optical networks. In *Proceedings of SPAA 2009*, pages 154–162, Calgary, 2009.

D. B. M. M. Fontes and J. F. Gonçalves. Heuristic solutions for general concave minimum cost network flow problems. *Networks*, 50:67–76, 2007.

B. Fortz and M. Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29:13–48, 2004. Preliminary short version of this paper published as "Internet Traffic Engineering by Optimizing OSPF weights," in Proc. IEEE INFOCOM 2000 – The Conference on Computer Communications.

M. Gendreau and J.-Y. Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, 2nd edition, 2010.

F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

J. F. Gonçalves. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European J. of Operational Research*, 183:1212–1229, 2007.

J. F. Gonçalves and J. Almeida. A hybrid genetic algorithm for assembly line balancing. *J. of Heuristics*, 8:629–642, 2002.

J. F. Gonçalves and N. C. Beirão. Um algoritmo genético baseado em chaves aleatórias para sequenciamento de operações. *Revista Associação Portuguesa de Desenvolvimento e Investigação Operacional*, 19:123–137, 1999.

J. F. Gonçalves and M. G. C. Resende. An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering*, 47:247–273, 2004.

J. F. Gonçalves and M. G. C. Resende. A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. Technical report, AT&T Labs Research, Florham Park, NJ 07733 USA, 2009. To appear in *J. of Combinatorial Optimization*.

J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European J. of Operational Research*, 167: 77–95, 2005.

J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. *European J. of Operational Research*, 189:1171–1190, 2008.

J. F. Gonçalves, M. G. C. Resende, and J. J. M. Mendes. A biased random-key genetic algorithm with forward-backward improvement for the resource

constrained project scheduling problem. *J. of Heuristics*, 2010. URL `http://dx/doi/org/10.1007/s10732-010-9142-2`.

J.F. Gonçalves and M.G.C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 2010a. doi: 10.1007/s10732-010-9143-1.

J.F. Gonçalves and M.G.C. Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. Technical report, AT&T Labs Research, 2010b. URL `http://www.research.att.com/~mgcr/doc/brkga-pack3d.pdf`.

J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36:92–109, 2009.

T. F. Noronha and C. C. Ribeiro. Routing and wavelength assingn by partition coloring. *European Journal of Operational Research*, 171:797–810, 2006.

T. F. Noronha, M. G. C. Resende, and C. C. Ribeiro. A biased random-key genetic algorithm for routing and wavelength assignment. *J. of Global Optimization*, 2010. doi: 10.1007/s10898-010-9608-7.

R. Reis, M. Ritt, L. S. Buriol, and M. G. C. Resende. A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion. *International Transactions in Operational Research*, 2010. doi: 10.1111/j.1475-3995.2010.00771.x.

M. G. C. Resende, R.F. Toso, J. F. Gonçalves, and R. M. A. Silva. A biased random-key genetic algorithm for the Steiner triple covering problem. Technical report, AT&T Labs Research, October 2010.

M.G.C. Resende and P.M. Pardalos, editors. *Handbook of Optimization in Telecommunicatoins*. Springer, 2006.

M.G.C. Resende, R.F. Toso, J.F. Gonçalves, and R.M.A. Silva. A biased random-key genetic algorithm for the steiner triple covering problem. *Optimization Letters*, 2011. To appear.

N. Skorin-Kapov. Routing and wavelength assignment in optical networks using bin packing based algorithms. *European Journal of Operational Research*, 177:1167–1179, 2007.

W. M. Spears and K. A. DeJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

J. M. S. Valente and J. F. Gonçalves. A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. *Computers and Operations Research*, 35:3696–3713, 2008.

J. M. S. Valente, J. F. Gonçalves, and R. A. F. S. Alves. A hybrid genetic algorithm for the early/tardy scheduling problem. *Asia-Pacific J. of Operational Research*, 23:393–405, 2006.

(Mauricio G.C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.

E-mail address: `mgcr@research.att.com`