*Engineering Optimization*
Vol. 36, No. 2, April 2004, 189–205

Taylor & Francis
Taylor & Francis Group

# CAPACITATED NETWORK DESIGN CONSIDERING SURVIVABILITY: AN EVOLUTIONARY APPROACH

ABDULLAH KONAK[a,*] and ALICE E. SMITH[b,†]

[a]*Information Sciences and Technology, Penn State Berks-Lehigh Valley, Tulpehocken Road, P.O. Box 7009, Reading, PA 19610-6009, USA;* [b]*Department of Industrial and Systems Engineering, Auburn University, 207 Dunstan Hall, Auburn, AL 36849-5346, USA*

This paper presents an evolutionary approach to design of capacitated networks considering cost, performance, and survivability. Traditionally, network performance and survivability have been considered independently. The approach presented in this paper is comprehensive where selecting network topology, assigning capacities for each link, and determining a route for each communicating node pair are simultaneously performed during optimization. Dual objectives of minimizing cost and minimizing delay are used, and the network design is subject to a survivability constraint. The proposed approach is tested on problems from the literature, and it is shown that it improves significantly upon previous single objective approaches and provides the user with a Pareto optimal set of designs to examine further.

*Keywords*: Network design; Capacitated networks; Survivability; Multi-objective design

## 1 INTRODUCTION

In this paper, an evolutionary algorithm (EA) is proposed to solve the survivable capacitated network design problem, which frequently arises in the design of telecommunications and computer networks. The capacitated network design problem (CNDP) is briefly defined as follows. A complete undirected network $G = (V, E)$ with node set $V = \{1, 2, \ldots, n\}$ and link set $E = \{(i, j)\colon i, j \in V, i \neq j\}$ is given. $K$ commodities, each representing node-to-node traffic, are simultaneously routed on the network; each commodity $k$ has a single source node, $s(k)$, a single destination node, $d(k)$, and a traffic amount of $t(k)$. It is assumed that all traffic from node $s(k)$ to node $d(k)$ is sent along a single path (*i.e.* network flows are non-bifurcated). There are $L$ link types available to install on the network. Let $Q_l$ denote the capacity of link type $l$ such that $Q_1 < Q_2 < \cdots < Q_L$. If link type $l$ is installed between nodes $i$ and $j$, a construction of cost $c_{ij}^l$ is incurred. The total design cost of the network is given by

$$Z_C = \sum_{(i,j)\in E, i<j} \sum_{l=1,\ldots,L} c_{ij}^l x_{ij}^l, \tag{1}$$

where $x_{ij}^l = 1$ if link type $l$ is used between nodes $i$ and $j$ and $x_{ij}^l = 0$, otherwise.

* E-mail: konak@psu.edu
† Corresponding author. E-mail: aesmith@eng.auburn.edu

*Engineering Optimization*
ISSN 0305-215X print; ISSN 1029-0273 online 
http://www.tandf.co.uk/journals
DOI: 10.1080/03052150310001633223

The routing objective is to minimize the average packet delay during peak hours, which is the average time that a packet spends in the network. The average packet delay is usually measured using the Kleinrock delay function [1] given by

$$Z_T = \frac{1}{\gamma} \sum_{(i,j)\in E, i<j} \frac{f_{ij}}{\lambda u_{ij} - f_{ij}}, \tag{2}$$

where $f_{ij}$ is the packet arrival rate (pps) to link $(i, j)$, $u_{ij}$ is the link capacity (bps), $1/\lambda$ is the average packet length (bpp), and $\gamma$ denotes the total number of the packets in the network, *i.e.*, $\gamma = \sum_{k=1}^{K} t(k)$. The derivation of Eq. (2) is based on several assumptions – packet arrivals are Poisson distributed, packet sizes are exponentially distributed, there is infinite buffer capacity, and the first-come first-served rule is used at nodes to route packets. Therefore, Eq. (2) is only an approximation to the actual packet delay of a network. However, it has been found that even though $Z_T$ is quite different from the actual packet delay, minimizing it in the design process, or during routing, will lead to good network performance in implementation [2]. Therefore, Eq. (2) has been extensively used in the literature as a measure of network performance.

The CNDP has three sets of decision variables: (i) the network topology (which nodes should be directly connected), (ii) link capacities (which link types should be installed on connections), and (iii) routes (how the traffic between each node pair should be routed). The CNDP is an extremely difficult problem mainly due to the fact that a large number of integer variables are involved in the mathematical models and linear programming relaxation does not provide tight bounds. In addition, even finding a feasible solution for the problem is quite difficult. Therefore, several heuristic approaches have been applied to the problem, including the branch-exchange method [3], the concave branch elimination method [4], the cut-saturation algorithm [5, 6], construction heuristics [7], simulated annealing [8], tabu search [9, 10], genetic algorithms [11], and expert systems [12, 13]. In addition, methods with performance guarantees have been developed based on Lagrangian relaxation of the mathematical formulation [14–17]. The common approach applied in these methods is to relax the coupling constraints or the flow conservation constraints, then separate the problem into subproblems that can be efficiently solved. Lagrangian based relaxation has been known to provide good results when the topology of the network is limited to a tree [18] or a ring [19] and when a fixed network topology is considered [20–23]; then, the CNDP is reduced to the capacity and route assignment problem.

In addition to cost and performance, another important consideration in network design is survivability. Network survivability describes the ability of a network to restore services after component failures, such as catastrophic failures of links and nodes. Network survivability is an important concern as network topologies are becoming sparse because of new high capacity fiber-optic links and technologies enabling few links to carry all of the network traffic. When a single link fails in a high-capacity sparse network with few diverse paths between nodes, significant service disturbance occurs or the network can lose its global connectivity. Traditionally, the survivability of a network against component failures is achieved by imposing redundant paths between nodes. For many real-life network applications, a level of redundancy that provides connectivity in case of a single link or node failure is sufficient since component failures are so rare that the probability of observing another failure during the repair of a failed component is almost zero [24, 25]. Therefore, researchers mainly focus on 2-link and 2-node network connectivity problems, which assure network connectivity against a single component failure. A network is said to be 2-link (2-node) connected if there exist 2-link disjoint (2-node disjoint) paths between every node pair. Previous approaches to the survivable network design problem (SNDP) include construction and improvement heuristics [42], genetic algorithms [26–30], tabu search [31], and integer programming [24, 32–34].

Unfortunately, the majority of the work addressing network survivability ignores performance. The SNDP is usually formulated so as to design a minimum cost, uncapacitated network topology satisfying a specified connectivity. Likewise, the CNDP research focusing on performance and cost ignores survivability, or considers simple connectivity measures, such as node degrees. Only a few papers address the survivable network design problem considering a performance measure [6, 10, 12, 35, 36].

All network design problems discussed in the previous section are known to be NP-hard. Multicommodity network flow problems combining topology and route selection decisions, such as the CNDP, are known to be extremely difficult even for small problem instances. Considering both survivability and performance makes the problems much harder as the mixed-integer formulations become very large. Another difficulty of considering survivability and performance together is that survivability constraints destroy the separable structure of the problem, which is exploited by Lagrangian relaxation approaches to solve large-scale network design problems. Similarly, considering network flows makes it impossible to use the same approaches used to solve the SNDP. The research presented in this paper proposes an EA to solve the CNDP with 2-node connectivity constraints, incorporating the CNDP and the SNDP into one problem, the survivable capacitated network design problem (SCNDP). In this problem, node connectivity is used because the processing units of a network (nodes) are more likely to fail than the static connections (links).

In the literature, the CNDP is usually formulated as a single objective design problem, minimizing $Z_C$ for a given maximum $Z_T$, or vice versa. In reality, the CNDP is a dual objective design problem with an obvious trade-off between cost and performance. Designers would prefer to investigate the trade-off curve of the cost and performance before choosing a final design. In addition, $Z_T$ is actually a surrogate measure for the actual performance of a network. Therefore, it is mainly helpful to compare alternative designs, rather than operating as a constraint. Therefore, in this paper, the objectives of minimizing cost and minimizing the packet delay are considered concurrently.

## 2 APPROACH AND ALGORITHM

### 2.1 Background

Evolutionary algorithms is a broad term used to refer to a group of meta-heuristic optimization approaches (algorithms) that imitate natural evolution in order to solve difficult optimization and design problems. Several evolutionary approaches have been proposed, such as genetic algorithms, evolutionary programming, and evolution strategies after the pioneering work by Holland [37] in the 1970s. All EA approaches operate on a set of solutions called a *population*. The population is continuously modified during the search process by adding new solutions and removing existing ones. New solutions are produced by two means, *crossover* and *mutation*. Imitating breeding, crossover combines two or more solutions (called *parents*) from the population in order to produce new solutions (called *offspring*). Mutation produces new solutions by small perturbations in the structure of a solution. Solutions are removed from the population by a selection mechanism which takes the relative quality of solutions into consideration. The quality of the solutions is assessed by evaluating and assigning them scalar *fitness* values based on the decision-maker's objectives.

In a multi-objective design problem, the decision-maker has two or more conflicting objectives, meaning that one of them cannot be improved without aggravating others. In fact, many real-life engineering design problems have several conflicting objectives, making a multi-objective approach advantageous. A multi-objective design problem is formally defined

as follows: given a set of $n$ design parameters $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, find a design $\boldsymbol{x}^*$ that optimizes $k$ objective functions $\boldsymbol{f}(\boldsymbol{x}^*) = \{f_1(\boldsymbol{x}^*), \ldots, f_k(\boldsymbol{x}^*)\}$ subject to $g(\boldsymbol{x}^*) = \boldsymbol{b}$. Since the design objectives conflict with each other, to find a perfect design that simultaneously optimizes each objective function is impossible. A reasonable approach to a multi-objective design problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. Assuming that all objectives are of the minimization type, a solution $\boldsymbol{x}$ is said to dominate another solution $\boldsymbol{y}$ if, and only if, $\boldsymbol{f}_i(\boldsymbol{x}) \leq \boldsymbol{f}_i(\boldsymbol{y})$ for $i = 1, \ldots, k$ and $\boldsymbol{f}_j(\boldsymbol{x}) < \boldsymbol{f}_j(\boldsymbol{y})$ for at least one objective. A solution is said to be *Pareto optimal* if it is not dominated by any other feasible solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective. The set of all non-dominated solutions is referred to as the *Pareto optimal set*, and for a given Pareto optimal set, the corresponding objective function values in the objective space are called the *Pareto front*. The ultimate goal of a multi-objective optimization algorithm is to identify solutions in the Pareto optimal set. In recent years, EA approaches have become very popular for multi-objective optimization problems. This popularity can be attributed to their inherent parallelism and capability to exploit the similarities of Pareto optimal solutions in order to generate new solutions by means of crossover. Interested readers should refer to the comprehensive survey papers [38–42] on multi-objective optimization using EA. A similar approach to the one in this paper is by Knowles and Corne [43] which describes a straightforward method to identify the Pareto front for evolutionary strategies.

## 2.2 Approach

The first step in the implementation of an EA approach is to choose an encoding scheme. Special care should be taken when choosing the encoding scheme since the performance of an EA highly depends on it. A pure EA approach to solve the SCNDP requires a complex encoding scheme to represent network topology, link capacities, and routes simultaneously. However, complex encoding schemes are known to have negative effects on performance [44]. The selected encoding represents only capacitated networks without routes, and for a given capacitated network, routes are determined by solving the routing problem. By dividing the problem, the encoding becomes simple.

Another important concern for an EA (or any heuristic method) is handling of constraints. Research has considered different ways to deal with constraints, such as special crossover and mutation operators that can always produce feasible solutions, repair algorithms, and penalty function methods [45]. Constraints such as 2-node connectivity are particularly difficult to deal with in EA. First, it is computationally expensive to evaluate the feasibility of solutions. Second, the degree of infeasibility cannot be clearly quantified for use in a penalty function. In addition, traditional EA operators such as single point crossover and bit flip mutation frequently produce disconnected or infeasible network topologies. Therefore, specialized encoding schemes and operators to produce feasible network topologies have been active research topics in the EA literature (see examples in [26, 28, 46, 47]).

In the SCNDP, a feasible solution must satisfy the 2-node connectivity constraints and must have enough capacity so that all traffic can be successfully delivered. To handle the 2-node connectivity constraints, a uniform crossover with an efficient repair algorithm and special mutation operators that can always produce 2-node connected networks are used. Therefore, a 2-node connectivity check is not required after mutation. To handle the infeasibility due to inadequate capacities, solutions are repaired by changing link capacities. The details of the crossover and mutation operators are given in the following sections.

### 2.3 Encoding

Networks are represented by an incidence matrix $X$ such that the $ij$th entry $x_{ij}$ is integer corresponding to the link type if link $(i, j)$ exists and 0 otherwise. For example, $x_{12} = 2$ means that link (1, 2) exists and it is a type 2 link. Since links are bi-directional only the upper half of $X$ is needed for the representation.

### 2.4 Initial Solutions

The EA starts with $\mu_0$ feasible solutions randomly generated in two steps. First, a random 2-node connected topology is constructed using the ear decomposition procedure [25] as follows. This procedure starts with a partial solution, which includes a cycle spanning randomly selected subsets of the nodes. In the following steps, random paths (called *ears*), starting and terminating at different nodes in the partial solution and traversing only the nodes that are not in the solution yet, are added to the solution until all nodes are included. To make sure that enough link capacity is installed on the network, first the traffic between each node pair is routed through the shortest path between them, and then a random link type is selected for each link such that links have enough capacity for the traffic flow through them.

### 2.5 Crossover Operator

In uniform crossover, an offspring randomly inherits genes (links) from either parent, one at a time. As a result of uniform crossover, it is possible that the offspring is not 2-node connected or not even connected at all. A repair algorithm is used to produce an acceptable topology as follows. Each node $v \in V$ is removed from the offspring one at a time. Then, starting from the smallest indexed node in $V \backslash v$ a depth-first search is performed to determine whether the nodes in $V \backslash v$ are connected. Let $S(v)$ be set of nodes in the search tree of the depth-first search and $\bar{S}(v)$ be the remaining nodes, *i.e.*, $\bar{S}(v) = V \backslash \{v \cup S(v)\}$. If $\bar{S}(v) = \emptyset$, then the offspring is 1-node fault-tolerant with respect to node $v$. If $\bar{S}(v) \neq \emptyset$, then removing $v$ disconnects the remaining nodes. To achieve connectivity, a link $(i, j)$ from either parent such that $i \in S(v)$ and $j \in \bar{S}(v)$ is added to the offspring. If more than one link satisfies the condition, then the one with the minimum cost is chosen. After adding link $(i, j)$, the search tree is updated and the depth-first search continues until $\bar{S}(v) = \emptyset$. The rationale for this repair algorithm is that both parents are 2-node connected; therefore, both parents must have at least one link $(i, j)$ connecting nodes in $S(v)$ and $\bar{S}(v)$ when node $v$ is removed.

Using the connectivity check/repair algorithm, the 2-node connectivity of a network can be determined in $O(nm)$ time (the depth-first search requires $O(m)$ time and it is invoked $n$ times, once for each node). If the offspring is not 2-node connected, the repair operation requires only a few additional steps since the connectivity check and repair are performed simultaneously.

The connectivity check/repair algorithm is computationally expensive. However, it can be avoided in cases where the topology of one parent is a subset of the other's. In this case, uniform crossover always produces a 2-node connected offspring given that parents are 2-node connected. Therefore, a quick check of this relationship between parents prevents the necessity of the connectivity check/repair. The procedure of uniform crossover with the repair algorithm is given as follows:

*Procedure: Uniform Crossover with Repair*
Randomly and uniformly select two parents $X$ and $Y$ such that $X \neq Y$
For $i = 1, \ldots, n$ and $j = i + 1, \ldots, n$ do {
    If Uni[0, 1] $>$ 0.5 then $z_{ij} = x_{ij}$ else $z_{ij} = y_{ij}$
}

```
If (X ⊆ Y) ∨ (Y ⊆ X) return Z
//connectivity test and repair algorithm//
For each node v ∈ V do {
        Remove node v from offspring Z
        Let s be the node with the smallest index in V \ v
        S(v) = {s}, LIST = {s}, and S̄(v) = V \ s
        While S̄(v) ≠ Ø do {
                While LIST ≠ Ø do {
                        Select node i from the end of LIST
                        If there exists a link (i, j) such that j ∈ S̄(v) then {
                                Remove node j from S̄(v) and add to S(v)
                                Add node j to the end of LIST
                        }       else Delete node i from LIST
                }
                If S̄(v) ≠ Ø then        {
                Find a link (i, j) such that min{c_ij: i ∈ S(v), j ∈ S̄(v), and x_ij + y_ij ≥ 1}
                z_ij = Uni_Int[1, L]
                Remove node j from S̄(v) and add to S(v)
                Add node j to the end of LIST
                }
        }
}
return Z
```

### 2.6 Mutation Operators

There are several mutation operators perturbing network topologies without disturbing 2-node connectivity. To achieve this, the mutation operators create a solution from an existing solution by changing the cycles of the solution. Every 2-node connected network must include at least one cycle, and replacing a cycle with other cycles of the same nodes does not disturb the 2-node connectivity of the network. This property is used in Ref. [25] to construct local move operators for improvement heuristics to design minimum cost 2-node and 2-link connected network topologies. The mutation operators are given below.

*Procedure: One-Link Exchange*

Find a random cycle $C$ of at least four nodes.
Randomly choose four nodes $a$, $b$, $c$, and $d$ of the cycle such that $x_{ab} \geq 1$, $(a, b) \notin C$, and $x_{cd} = 0$.
Set $x_{cd} = x_{ab}$ and $x_{ab} = 0$.

End

*Procedure: Two-Link Exchange*

Find a random cycle $C$ of at least four nodes.
On cycle $C$, randomly determine 2 links $(a, b)$ and $(c, d)$ such that $x_{ab} \geq 1$, $x_{cd} \geq 1$, $x_{ac} = 0$, and $x_{bd} = 0$.
Set $x_{ac} = x_{ab}$, $x_{bd} = x_{cd}$, $x_{ab} = 0$, and $x_{cd} = 0$.

End

*Procedure: Three-Link Exchange*

Find a random cycle $C$ of at least four nodes.

On cycle $C$, randomly choose 3 links $(a, b)$, $(c, d)$, and $(e, f)$ such that $x_{ab} \geq 1$, $x_{cd} \geq 1$, $x_{ef} \geq 1$, $x_{ad} = 0$, $x_{be} = 0$, and $x_{cf} = 0$.
Set $x_{ad} = x_{ab}$, $x_{be} = x_{ef}$, $x_{cf} = x_{cd}$, $x_{ab} = 0$, $x_{cd} = 0$ and $x_{ef} = 0$.
End

*Procedure: Delete-a-Link*
Find a random cycle $C$ of at least four nodes.
Randomly choose 2 nodes $a$ and $b$ such that $x_{ab} \geq 1$ and $(a, b) \notin C$.
Delete link $(a, b)$.
End

*Procedure: Add-a-Link*
Randomly choose 2 nodes $a$ and $b$ such that $x_{ab} = 0$.
Set $x_{ab} =$ Uni_Int$[1, L]$.
End

*Procedure: Change-Link-Type*
Randomly choose a link $(a, b)$ such that $x_{ab} \geq 1$
Change type of link $(a, b)$ to another type.
End

### 2.7 Determining Routes

As mentioned earlier, the CNDP has three sets of decision variables: network topology, link types (capacities), and routes. Each encoding represents the network topology and link capacities. Once network topology and link capacities are determined and the connectivity constraints are satisfied, the design problem reduces to the routing problem, as follows:

$$\text{minimize} \quad Z_T = \frac{1}{\gamma} \sum_{(i,j)\in E, i<j} \frac{f_{ij}}{\lambda u_{ij} - f_{ij}} \tag{3}$$

subject to :

$$\sum_{\{i:(i,j)\in E\}} z_{ij}^k - \sum_{\{i:(j,i)\in E\}} z_{ji}^k = \begin{cases} 1 & \text{if } i = s(k) \\ -1 & \text{if } i = d(k) \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in V, \ \ k = 1, \ldots, K \tag{4}$$

$$\begin{aligned} &\sum_{k=1,\ldots,K} t(k)(z_{ij}^k + z_{ji}^k) = f_{ij} && \text{for all } (i, j) \in E, \ \ i < j \\ &z_{ij}^k \in \{0, 1\} && \text{for all } (i, j) \in E, \ \ k = 1, \ldots, K \\ &f_{ij} \geq 0 && \text{for all } (i, j) \in E, \ \ i < j. \end{aligned} \tag{5}$$

The mathematical formulation given above is an integer multicommodity network flow problem with a nonlinear objective function. Although the routing problem is much simpler than the overall design problem, it is still NP-hard due to non-bifurcated flows, *i.e.* traffic between a node pair is to be sent through a single path which is determined by binary route selection variables $z_{ij}^k$. However, this problem has been widely studied in the literature and several approaches have been developed to efficiently solve it, including the flow deviation method [1], Lagrangia nbased relaxation [20–23, 48], decomposition [49], and genetic algorithms [50, 51].

In this research, the flow deviation method [1] is used since it can quickly find or get very close to the global optimal flow in relatively few iterations [52, 53]. This method is known to perform particularly well on networks with a large number of nodes and relatively balanced traffic [1]. In addition, it is computationally easier to implement than other approaches. The flow deviation method starts with an initial feasible flow. In each iteration, a new flow is obtained by deviating a fraction of the current flow from existing paths to the shortest paths, which are determined by using the routing costs as calculated by taking the partial derivative of the objective function given in Eq. (2) with respect to the current flow. The algorithm terminates when the improvement in average delay becomes negligible.

*Procedure: Flow Deviation Method*
//finding an initial flow//
For each link $(i,\ j)$, set marginal cost $w_{ij} = \frac{1}{\gamma} \cdot \frac{1}{\lambda\, u_{ij}}$
For $k = 1, \ldots,\ K$ do {
    Find the shortest path $SP(k)$ from $s(k)$ to $d(k)$ using marginal link cost $w_{ij}$s
    Send $t(k)$ unit traffic through path $SP(k)$ and update flows on the path.
    Update the marginal costs of the links on $SP(k)$ as $w_{ij} = \frac{1}{\gamma} \cdot \frac{\lambda\, u_{ij}}{(\lambda\, u_{ij} - f_{ij})^2}$.
    }
//improve the current flow//
Do {
    For $k = 1, \ldots, K$ do {
        Find the new shortest path $NSP(k)$ from $s(k)$ to $d(k)$ ( $NSP(k)$) using $w_{ij}$s.
        Route commodity $k$ through path $NSP(k)$ and update the flow.
        If the new flow is feasible and improves $Z_T$, $SP(k) = NSP(k)$,
        otherwise keep the current route.
    }
    Calculate new marginal link costs as $w_{ij} = \frac{1}{\gamma} \cdot \frac{\lambda\, u_{ij}}{(\lambda\, u_{ij} - f_{ij})^2}$.
} While (A new path improving $Z_T$ is found)
End

It is possible that an initial feasible flow cannot be determined for a solution. In this case, the link capacities of the solution are adjusted as shown below instead of discarding the entire solution.

*Procedure: Repair Link Capacities*
For $k = 1, \ldots, K$ do {
    Find the shortest path $SP(k)$ from $s(k)$ to $d(k)$
    Sent $t(k)$ unit traffic through the shortest path $SP(k)$
    Update the network flow
    }
Determine new link types for each $(i,\ j)$ such that $x_{ij} = \min\{l\colon Q_l > f_{ij}\}$
Call the Procedure Flow Deviation to improve the flow to
minimize delay
End

### 2.8 Overall Algorithm

The algorithm keeps Pareto optimal solutions found during the search until better solutions are found. In a comparative study [52], Zitzler *et al*. report that this elitist approach outperforms other approaches such as objective weighting and Pareto ranking. This approach also simplifies

the implementation as no secondary population is needed to store all Pareto front solutions. In addition, it promotes a diverse Pareto front since all Pareto front solutions contribute to the search as opposed to the case with two populations where only the solutions of the primary population participate in the search through crossover and mutation while the secondary population is used as an archive of Pareto front solutions.

In each iteration, a single solution is generated by either crossover or mutation, selected randomly uniformly, an incremental EA. In mutation, one of the mutation operators is also randomly and uniformly selected. The new solution is compared to all solutions in the population, and is discarded if it is dominated by any existing solution. Any existing solutions dominated by the new solution are removed. The solution is added to the population if none of these cases occur. Therefore, the size of the population dynamically changes during the search and it is possible that it might grow too large. To prevent this, the size of the population ($\mu$) is controlled by two parameters, maximum and minimum population size ($\mu_{\max}$ and $\mu_{\min}$, respectively) for the larger problems (20 nodes and above). When $\mu$ reaches $\mu_{\max}$, $\mu_{\max} - \mu_{\min}$ solutions are removed from the population giving consideration to maintaining the diversity of the Pareto front. To achieve this, a tournament selection based on niching [41, 54] is used as follows. Two solutions are randomly chosen and the solution with the higher niche count is removed. The niche count of solution $X$ is given by

$$nc(X) = \sum_{Y \in P} \max \left\{ \frac{\sigma_{\text{share}} - d(X, Y)}{\sigma_{\text{share}}}, 0 \right\}, \tag{6}$$

where $\sigma_{\text{share}}$ is the niche size and $d(X, Y)$ is the Euclidean distance between solutions $X$ and $Y$ calculated in the normalized objective space between 0 and 1, *i.e.*

$$d(X, Y) = \sqrt{\left( \frac{Z_C(X) - Z_C(Y)}{Z_C^{\max} - Z_C^{\min}} \right)^2 + \left( \frac{Z_T(X) - Z_T(Y)}{Z_T^{\max} - Z_T^{\min}} \right)^2}, \tag{7}$$

where $Z_T^{\min}$ and $Z_C^{\max}$ are the delay and cost of the most expensive solution and $Z_T^{\max}$ and $Z_C^{\min}$ are the delay and cost of the least expensive solution in the population, respectively. The niche size defines a neighborhood of solutions in the objective space. The solutions sharing the same neighborhood contribute to each other's niche count. Therefore, a solution with a crowded neighborhood will have a higher niche count, reducing the chance of that solution surviving. As a result, niching limits the proliferation of solutions in one particular neighborhood of the objective space. In fact, niching is a widely used approach in multi-objective GAs to make sure that a diverse Pareto front is obtained, and has many variants (see Ref. [42]).

Iterations continue until no new solution improving on a current Pareto front solution is found during the last $g_{\text{stop}}$ solutions evaluated or a maximum number of new solutions ($g_{\max}$) is reached. As a final note on the method, like other EA, this method has a number of parameters to be set prior to optimization. The performance of the method, however, is not sensitive to the exact values of these parameters and the search algorithm performs quite robustly over a range of them. Therefore, the designer should not have to spend much time choosing parameter values for a given problem instance.

## 3 COMPUTATIONAL EXPERIMENTS

Four test problems taken from Ref. [10] were studied. These test problems are 10-, 15-, 20-, and 23-node networks and eight link types are available in each problem. The given data of the

problems are solved to include the Cartesian coordinates of the nodes, the traffic load between each node pair, the average packet size, the possible link options in terms of capacity, the fixed and variable costs of the link options, and the routing policy. In Ref. [10], the problems are solved to minimize the design cost for a given maximum allowable delay and node degree constraints, and single objective TS is used as the solution approach. In addition, comparative results are presented for simulated annealing, a genetic algorithm, and the cut saturation algorithm. Since in the research herein 2-node connectivity is used, direct comparison of results with Ref. [10] could be misleading. In a 2-node connected network each node must have at least two links. Therefore, each solution found herein is a feasible solution with respect to a node degree requirement of two. However, the opposite case is not true, *i.e.* a network with minimum node degree of two may not be 2-node connected. This implies that the problem formulated herein is more constrained than the problems in Ref. [10]. In addition, the connectivity check for 2-node connected networks is computationally more expensive than a check for node degrees.

After initial experimentation, the following parameters were used for all problems: $\mu_0 = 20$, $\mu_{\min} = 400$, $\mu_{\max} = 600$, $g_{\max} = 500{,}000$, $g_{\text{stop}} = 3000$, and $\sigma_{\text{share}} = 0.05$. All runs were performed on a PC running on the LINUX operating system with 2 GHz CPU and 1 GB memory.

Figures 1–4 depict the initial population and the final population (the Pareto front) found for each problem. In addition, the previous results reported in Ref. [10] are depicted. In each case, the algorithm herein found many solutions dominating previous results. Particularly for the 20- and 23-node problems, the solutions were decidedly superior to previous results. This performance improvement for larger networks can be explained by the fact that flow deviation method performs especially well in large networks.

These significant improvements can be also attributed to the differences between the routing procedures used in this research and the previous ones. In Refs. [8, 10, 11] a simple routing and capacity assignment policy is used: each commodity is routed through shortest paths, link flows are calculated, and link capacities are determined by assigning the smallest available capacity option that is greater than the flow on each link. To demonstrate the performance

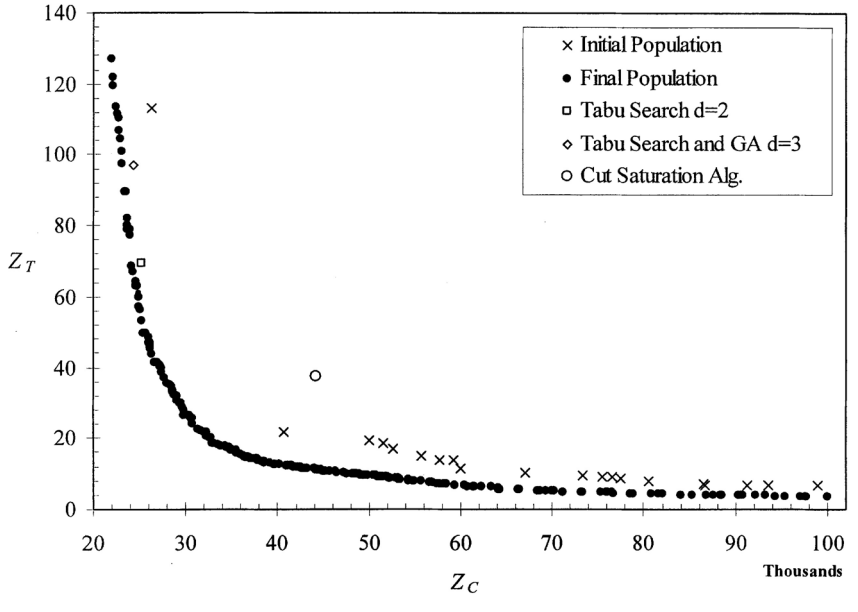


FIGURE 1 The Pareto front for the 10-node problem.

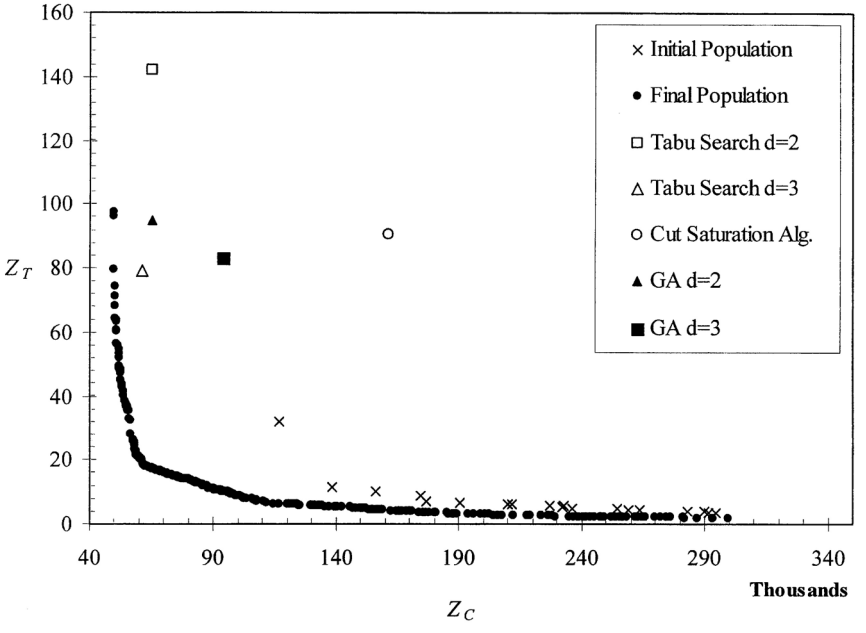


FIGURE 2 The Pareto front for the 15-node problem.

of the EA approach independent from routing method, it was run with the route and capacity assignment procedure used in Refs. [8, 10, 11]. Results are shown in Figures 5 and 6 for 10- and 23-node problems, respectively. In this case, the EA approach still found many solutions dominating previous results. Interestingly, some of the initial solutions found by the EA approach dominated previously-reported, single objective results. The likely explanation for this is that the ear decomposition procedure used to generate initial solutions is known to find low-cost,

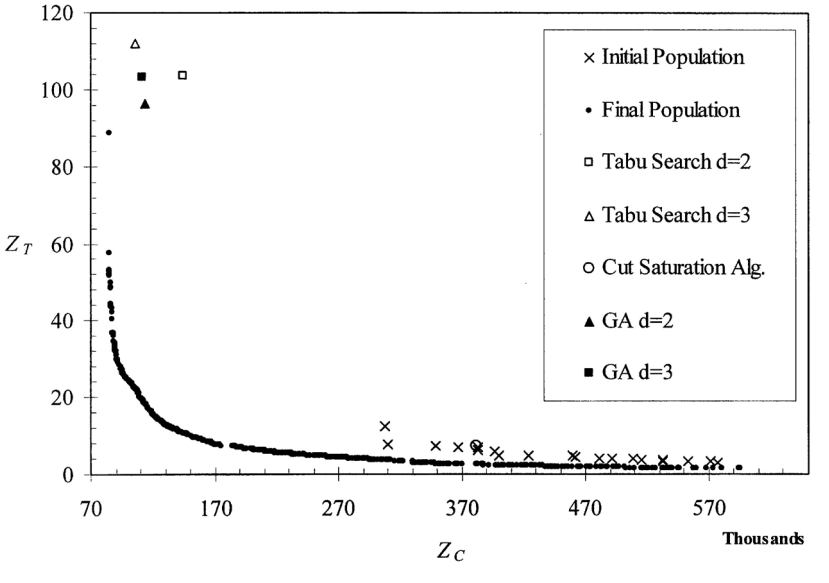


FIGURE 3 The Pareto front for the 20-node problem.

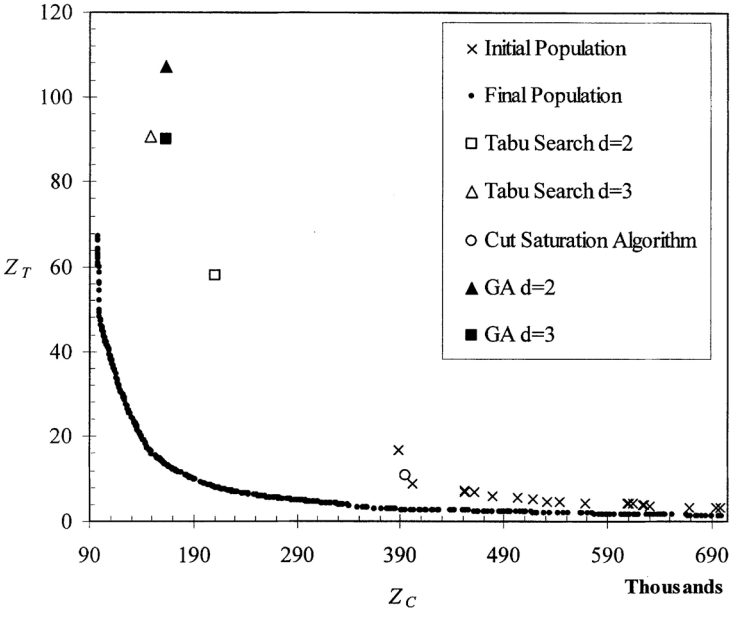


FIGURE 4 The Pareto front for the 23-node problem.

survivable networks with high route diversity (see Ref. [25] for further discussion), which leads to efficient routing. In other words, the initial solutions generated are apt to be quite good.

The main difference between using the flow deviation method and the shortest path routing is in the final Pareto fronts. In the former case, a wide Pareto front over which solutions are almost evenly scattered is identified, whereas the Pareto front in the latter case was confined to a smaller region towards low cost. This was a result of assigning the smallest feasible

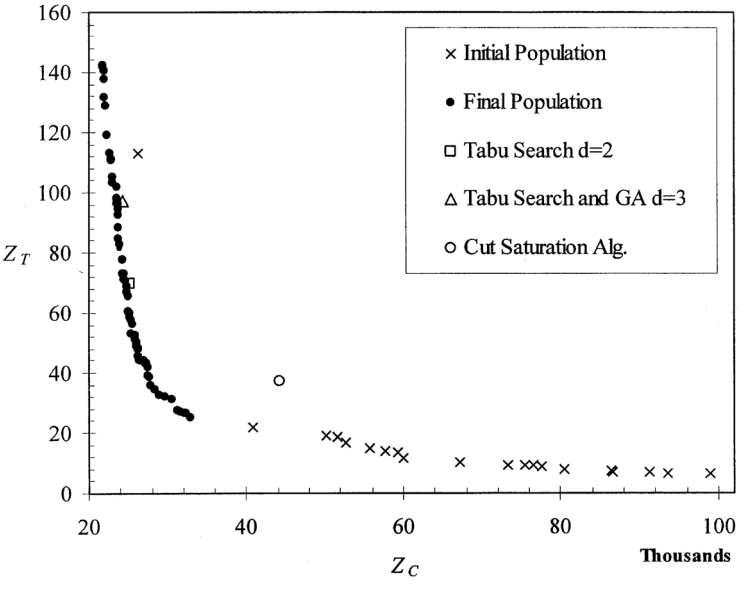


FIGURE 5 The Pareto front for the 10-node problem using the shortest path routing policy.

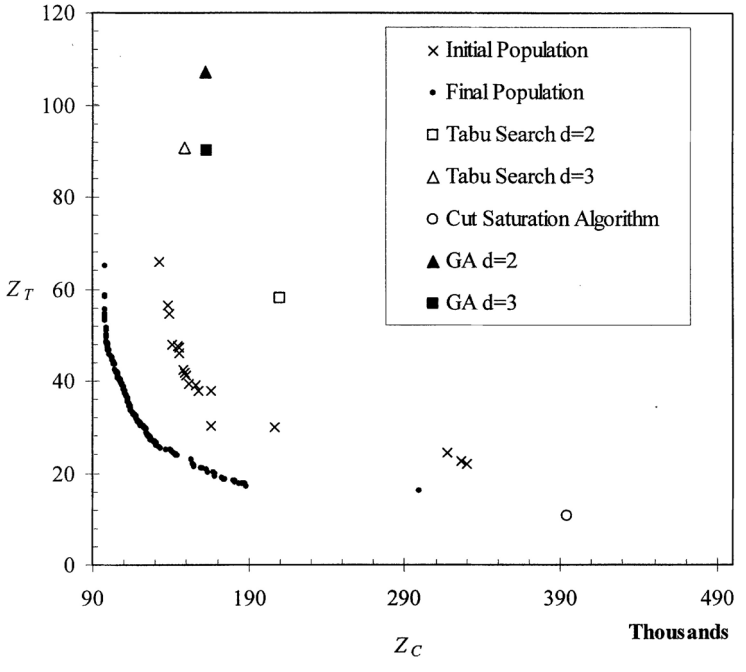


FIGURE 6 The Pareto front for the 23-node problem using the shortest path routing policy.

capacity option on each link. However, the Pareto front in the former case dominates the one in the latter case, especially for high-cost solutions. This difference was also exhibited in the triggering of the maximum population size and niching mechanism. The only instance in which the maximum population size was reached was for the 23-node problem using flow deviation routing, where the population was pruned three times. This is due to the large number of Pareto optimal solutions identified for this instance.

Table I gives the number of solutions searched and the average CPU seconds per solution evaluated for each problem studied. Comparing CPU times, it is clear that most of the CPU time was spent in solving the routing problem in the flow deviation case. However, it should be noted the CPU time is not a primary concern in network design problems since the design task is an offline process with considerable time allowance. For all problems, CPU times are well within practical limits considering computationally intense operations such as the 2-node connectivity check and traffic routing.

TABLE I The Number of Solutions Searched and CPU Effort per Solution.

| | *Flow deviation method routing* | | *Shortest path routing* | |
|---|---|---|---|---|
| *Problem size (n)* | *Solutions searched* | *CPU seconds per solution* | *Solutions searched* | *CPU seconds per solution* |
| 10 | 500,000 | 0.00327 | 49,961 | 0.00020 |
| 15 | 500,000 | 0.01231 | 114,555 | 0.00036 |
| 20 | 500,000 | 0.03876 | 285,811 | 0.00065 |
| 23 | 500,000 | 0.06289 | 429,392 | 0.00095 |

## 4 CONCLUSIONS AND DISCUSSION

The multi-objective survivable CNDP is a very difficult problem with frequent applications in telecommunication and computer networks. When network cost, network performance, and survivability are considered together in a network design problem, the resulting mathematical model become computationally intractable with an enormous number of decision variables and constraints. In addition, the problem is no longer separable, and efficient techniques exploiting this feature are no longer applicable.

The approach described in this paper divides the problem into two parts. The EA searches capacitated 2-node connected topologies and the relatively easier routing problem is then solved to minimize delay. Using this hybrid approach has several advantages. The EA is used to handle complex survivability constraints with tailored crossover and mutation operators, whilst the routing problem, for which it is difficult to develop an encoding scheme, is solved by using a well-known heuristic from the literature. Such hybrid methods, *i.e.* using evolutionary approaches to simplify the structure of a complex design problem, and then using exact or heuristic methods to efficiently solve sub problems, are promising. Hybrid methods will be useful in many complicated engineering design problems such as facilities design, reliability, and scheduling.

Further enhancing the method is the concurrent consideration of the two most common objectives, minimizing network cost and minimizing packet delay (that is, maximizing performance). By searching over the Pareto front of these conflicting objectives, network designers are presented with a set of superior designs so that they may explicitly examine the trade-off between cost and performance. Although using two objectives in the search is slightly more complicated than using a single objective (variable population size, comparison for dominance among solutions), it is still quite computationally tractable within an EA framework and identifies a wide range of excellent designs.

## NOMENCLATURE

| | |
|---|---|
| EA | evolutionary algorithms |
| CNDP | capacitated network design problem |
| SNDP | survivable network design problem |
| SCNDP | survivable capacitated network design problem |
| $G = (V, E)$ | undirected network with node set $V$ and link set $E$ |
| $n$ | number of nodes, $\|V\|$ |
| $m$ | number of links, $\|E\|$ |
| $(i, j)$ | undirected link $(i, j)$ between nodes $i$ and $j$ $((i, j) \equiv (j, i))$ |
| $Z_C$ | design cost of a network |
| $Z_T$ | average packet delay of a network (seconds) |
| $f_{ij}$ | amount of traffic flow through link $(i, j)$ (packets per second) |
| $u_{ij}$ | capacity of link $(i, j)$ (bits per second) |
| $1/\lambda$ | average packet length (bits per packet) |
| $\gamma$ | total number of packets fed into a network |
| $L$ | number of link types |
| $Q_l$ | capacity of link type $l$ |
| $K$ | number of commodities to be routed in the network |
| $s(k)$ | source node of commodity $k$ |

| | |
|---|---|
| $d(k)$ | destination node of commodity $k$ |
| $t(k)$ | traffic demand of commodity $k$ (packets per second) |
| $x_{ij}$ | type of link installed on link $(i, j)$, $x_{ij} = 0, \ldots, L$ such that $x_{ij} = 0$ denotes no link |
| $c_{ij}^{l}$ | cost of installing link type $l$ between nodes $i$ and $j$ |
| $z_{ij}^{k}$ | indicator variable such that $z_{ij}^{k} = 1$ if link$(i, j)$ is used to route commodity $k$ and $z_{ij}^{k} = 0$, otherwise |
| $[S(v), \bar{S}(v)]$ | cut set of a network after removing node $v$ |
| $g_{\max}$ | the maximum number of solutions evaluated |
| $g_{\text{stop}}$ | stopping criteria |
| $\mu_{\max}, \mu_{\min}$ | the maximum and minimum population size, respectively |
| $\mu_{\text{o}}$ | initial population size |
| $\sigma_{\text{share}}$ | niche size |

## *References*


[1] Fratta, L., Gerla, M. and Kleinrock, L. (1973) The flow deviation method: An approach to store-and forward communication network design. *Networks*, **(3)**, 97–133.

[2] Hayes, J. F. (1984) *Modeling and Analysis of Computer Communications Networks*. Plenum Press, New York.

[3] Tanenbaum, A. S. (1981) *Computer Networks,* Prentice-Hall, Englewood Cliffs, N.J.

[4] Gerla, M. and Kleinrock, L. (1977) On the topological design of distributed computer networks. *IEEE Transactions on Communications*, **COM-25**(1), 48–60.

[5] Boorstyn, R. R. and Frank, H. (1977) Large-scale network topological optimization. *IEEE Transactions on Communications*, **COM-25**(1), 29–47.

[6] Newport, K. T. and Varshney, P. K. (1991) Design of survivable communications networks under performance constraints. *IEEE Transactions on Reliability*, **40**(4), 433–440.

[7] Kershenbaum, A., Kermani, P. and Grover, G. A. (1991) MENTOR: an algorithm for mesh network topological optimization and routing. *IEEE Transactions on Communications*, **39**(4), 503–513.

[8] Pierre, S., Hyppolite, M.-A., Bourjolly, J.-M. and Dioume, O. (1995) Topological design of computer communication networks using simulated annealing. *Engineering Applications of Artificial Intelligence*, **8**(1), 61–69.

[9] Crainic, T. G., Gendreau, M. and Farvolden, J. M. (2000) A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, **12**(3), 223–236.

[10] Pierre, S. and Elgibaoui, A. (1997) A tabu-search approach for designing computer-network topologies with unreliable components. *IEEE Transactions on Reliability*, **46**(3), 350–359.

[11] Pierre, S. and Legault, G. (1998). A genetic algorithm for designing distributed computer network topologies. *IEEE Transactions on Systems, Man and Cybernetics, Part B [Cybernetics]*, **28**(2), 249–258.

[12] Dutta, A. and Mitra, S. (1993) Integrating heuristic knowledge and optimization models for communication network design. *IEEE Transactions on Knowledge and Data Engineering*, **5**(6), 999–1017.

[13] Fahmy, H. I. and Douligeris, C. (1995) END: an expert network designer. *IEEE Network*, **9**(6), 18–27.

[14] Altinkemer, K. and Yu, Z. (1992) Topological design of wide area communication networks. *Annals of Operations Research*, **36**(1–4), 365–381.

[15] Chang, S.-G. and Gavish, B. (1993) Telecommunications network topological design and capacity expansion: Formulations and algorithms. *Telecommunication Systems – Modeling, Analysis, Design and Management*, **1**(2), 99–131.

[16] Gavish, B. (1992) Topological design of computer communication networks – the overall design problem. *European Journal of Operational Research*, **58**(2), 149–172.

[17] Magnanti, T. L., Mirchandani, P. and Vachani, R. (1995) Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, **43**(1), 142–157.

[18] Gavish, B. (1985) Augmented Lagrangian based algorithms for centralized network design. *IEEE Transactions on Communications*, **COM-33**(12), 1247–1257.

[19] Altinkemer, K. (1994) Topological design of ring networks. *Computers and Operations Research*, **21**(4), 421–431.

[20] Amiri, A. and Pirkul, H. (1997) Routing in packet-switched communication networks with different criticality classes of communicating node pairs. *Naval Research Logistics*, **44**(5), 485–505.

[21] Amiri, A. and Pirkul, H. (1999) Routing and capacity assignment in backbone communication networks under time varying traffic conditions. *European Journal of Operational Research*, **117**(1), 15–29.

[22] Gavish, B. and Neuman, I. (1986) Capacity and flow assignment in large computer networks. *Proceedings of IEEE INFOCOM '86. Fifth Annual Conference on Computers and Communications Integration Design, Analysis, Management (Cat. No. 86CH2284–8)*. Miami, FL, USA, pp. 275–284.

[23] Gavish, B. and Neuman, I. (1992) Routing in a network with unreliable components. *IEEE Transactions on Communications*, **40**(7), 1248–1258.
[24] Grötschel, M., Monma, C. L. and Stoer, M. (1995) Design of survivable networks. *Network Models*. Handbooks in Operations Research and Management Science, ((Eds). M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser), Elsevier Science, Amsterdam, pp. 617–672.
[25] Monma, C. L. and Shallcross, D. F. (1989) Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, **37**(4), 531–541.
[26] Cheng, S.-T. (1998) Topological optimization of a reliable communication network. *IEEE Transactions on Reliability*, **47**(3, pt.1), 225–233.
[27] Davis, L., Orvosh, D., Cox, A. and Qiu, Y. (1993) A genetic algorithm for survivable network design. *Proceedings of the Fifth International Conference on Genetic Algorithms*. Urbana-Champaign, IL, USA, pp. 408–415.
[28] Huang, R., Ma, J. and Hsu, D. F. (1997) A genetic algorithm for optimal 3-connected telecommunication network designs. *Parallel Architectures, Algorithms, and Networks, 1997. (I-SPAN '97) Proceedings*. Taipei, Taiwan, pp. 344–350.
[29] Konak, A. and Smith, A. E. (1999) A hybrid genetic algorithm approach for backbone design of communication networks. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 1813, Washington, DC, USA, pp. 1817–1823.
[30] Kulturel-Konak, S., Konak, A. and Smith, A. E. (2000) Minimum cost 2-edge-connected Steiner graphs in rectilinear space: an evolutionary approach. *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*. Vol. 101, La Jolla, CA, USA, pp. 97–103.
[31] Koh, S. J. and Lee, C. Y. (1995) A tabu search for the survivable fiber optic communication network design. *Computers and Industrial Engineering*, **28**(4), 689–700.
[32] Balakrishnan, A., Magnanti, T. L. and Mirchandani, P. (1998) Designing hierarchical survivable networks. *Operations Research*, **46**(1), 116–136.
[33] Grötschel, M., Monma, C. L. and Stoer, M. (1992) Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, **40**(2), 309–330.
[34] Grötschel, M., Monma, C. L. and Stoer, M. (1995) Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, **43**(6), 1012–1024.
[35] Newport, K. T. (1990) Incorporating survivability considerations directly into the network design process. *Proceedings IEEE INFOCOM'90. The Conference on Computer Communications. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration (Cat. No. 90CH2826–5)*, Vol. 211, San Francisco, CA, USA, pp. 215–220.
[36] Pierre, S. and Beaubrun, R. (2000) Integrating routing and survivability in fault-tolerant computer network design. *Computer Communications*, **23**(4), 317–327.
[37] Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.
[38] Deb, K. (1999) Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Journal of Evolutionary Computation*, **7**(3), 205–230.
[39] Fonseca, C. M. and Fleming, P. J. (1995) An overview of evolutionary algorithms in multiobjective optimization. *Journal of Evolutionary Computation*, **3**(1), 1–16.
[40] Fonseca, C. M. and Fleming, P. J. (1998) Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example. *IEEE Transactions on Systems, Man and Cybernetics, Part A [Systems and Humans]*, **28**(1), 38–47.
[41] Van Veldhuizen, D. A. and Lamont, G. B. (2000) Multiobjective evolutionary algorithms: analyzing the state-of-the-art. *Evolutionary Computation*, **8**(2), 125–147.
[42] Zitzler, E. and Thiele, L. (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**(4), 257–271.
[43] Knowles, J. D. and Corne, D. W. (2000) Approximating the nondominated front using the Pareto archived evolutionary strategy. *Evolutionary Computation*, **8**(2), 149–172.
[44] Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Mass.
[45] Coit, D. W., Smith, A. E. and Tate, D. M. (1996) Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal of Computing*, **8**(2), 173–182.
[46] Jung, S. and Moon, B.-R. (2002) Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidean TSP. *IEEE Transactions on Evolutionary Computation*, **6**(6), 557–565.
[47] Palmer, C. C. and Kershenbaum, A. (1995) An approach to a problem in network design using genetic algorithms. *Networks*, **26**(3), 151–163.
[48] Pirkul, H. and Amiri, A. (1994) Routing in packet switched communication networks. *Computer Communications*, **17**(5), 307–316.
[49] Cantor, D. G. and Gerla, M. (1974) Optimal routing in a packet-switched computer network. *IEEE Transactions on Computers*, **C-23**(10), 1062–1069.
[50] Mann, J. W. and Smith, G. D. (1996) A comparison of heuristics for telecommunications traffic routing. *Modern Heuristic Search Methods*, ((Eds.) V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith), John Wiley, New York, NY, pp. 235–253.
[51] Munetomo, M. (2000) The genetic adaptive routing algorithm. *Telecommunications Optimization: Heuristic and Adaptive Techniques*, ((Eds.) D. W. Corne, M. J. Oates and G. D. Smith), John Wiley, New York, NY, pp. 151–166.

[52] Courtois, P.-J. and Semal, P. (1981) An algorithm for the optimization of nonbifurcated flows in computer communication networks. *Performance Evaluation*, **1**(2), 139–152.
[53] Gragopoulos, I., Papapetrou, E. and Pavlidou, F.-N. (2000) Performance study of adaptive routing algorithms for LEO satellite constellations under self-similar and Poisson traffic. *Space Communications*, **16**(1), 15–22.
[54] Deb, K. and Goldberg, D. E. (1989) An investigation of niche and species formation in genetic function optimization. *Third International Conference on Genetic Algorithms*. San Mateo, CA, USA, pp. 42–50.