Two-Edge Disjoint Survivable Network Design Problem with Relays

Abdullah Konak, Sadan Kulturel-Konak, and Alice E. Smith

Abstract In this paper, we study the network design problem with relays considering network survivability. This problem arises in telecommunications and logistic systems. The design problem involves selecting two edge-disjoint paths between source and destination node pairs and determining the location of the relays in the network to minimize the network cost. A constraint is imposed on the distance that a signal or commodity can travel in the network without being processed by a relay. An efficient and effective meta-heuristic approach is proposed to solve large sized problems and compared with construction heuristics and an exact approach run for up to 24 hours using CPLEX.

Keywords: Network Design, Survivability, Relays, Genetic Algorithm

1 Introduction

The network design problem with relays (NDPR) is first introduced by Cabral et al. (2007) in the context of a telecommunication network design problem. In this paper, we study the two-edge connected network design problem with relays

Abdullah Konak
Information Sciences and Technology, Penn State Berks
Tulpehocken Rd. PO Box 7009, Reading, PA19610 USA
konak@psu.edu

Sadan Kulturel-Konak
Management Information Systems, Penn State Berks
Tulpehocken Rd. PO Box 7009, Reading, PA19610 USA
sadan@psu.edu

Alice E. Smith
Department of Industrial & Systems Engineering, Auburn University
3301 Shelby Center, Auburn, AL 36849 USA
smithae@auburn.edu

(2ECON-NDPR), which is an extension of the NDPR by considering survivability in the network design. A survivable network has the ability to restore network services after catastrophic edge or node failures. Network survivability is an important concern in telecommunication network design as network topologies are becoming sparse because of high capacity fiber-optic links which can transmit large amount of data. 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 to protect against a single edge or node failure is sufficient since component failures are so rare that the probability of observing another failure during a repair is almost zero (Monma and Shallcross 1989; Magnanti et al. 1995). Therefore, network design researchers mainly focus on two-edge and two-node connectivity problems, which assure network connectivity against a single component failure. In this paper, we consider two-edge connectivity (2ECON) in the context of NDPR. However, our approach can be extended to two-node connectivity (2NCON) as well.

The 2ECON-NDPR is briefly defined as follows. An undirected network $G = (V,E)$ with node set $V = \{1,2,\ldots,N\}$ and edge set $E = \{(i,j) : i,j \in V, i < j\}$ is given. Each edge (i,j) is associated with a cost of $c^e_{i,j}$ and a distance of $d_{i,j}$. $|K|$ commodities, representing point-to-point traffics, are simultaneously routed on the network; each commodity $k$has a single source node $s(k)$ and a single destination node $t(k)$. For each commodity k, two edge-disjoint paths from node $s(k)$ to node $t(k)$ are to be determined. The first path $(p_1(k))$ is dedicated as the primary path for the routing of commodity k, and the second path $(p_2(k))$ is reserved as a backup path in case path $p_1(k)$ fails. In other words, the connectivity between nodes $s(k)$ and $t(\mathrm{k})$ for each commodity k is expected to be one edge fault tolerant similar to the two-edge connected network design problem (Monma and Shallcross 1989). In the 2ECON-NDPR, however, an upper bound λ is imposed on the distance that commodity k can travel on a path from node $s(k)$ to node $t(k)$ without visiting a special node, which is called a relay. A relay may correspond to different facilities on real life networks. For example, in digital telecommunication networks, the signal can be transmitted only for a limited distance without losing its fidelity due to attenuation and other factors, and therefore, it has to be regenerated at certain intervals during its transmission. In this case, a relay corresponds to a telecommunication facility in which the signal is regenerated into its original form. Different from the classic multi-commodity network design problem, in the 2ECON-NDPR some nodes of the network have to be dedicated as relays in which the signal is regenerated. A fixed cost, c^v_i, is occurred when a relay is located at node i. The objective function of the 2ECON-NDPR is to minimize the network design cost while making sure that each commodity k can be routed from node $s(k)$ to node $t(k)$ through two edge-disjoint paths on each of which the distances between node $s(k)$ and the first relay, between any consecutive relays, and between node $t(k)$ and the last relay are less than the upper bound λ. A network flow based formulation of the 2ECON-NDPR on a directed network, $G = (V,E)$, where edge set E includes both edges (i,j) and (j,i) is given as follows:

Decision Variables:

$x_{i,j}$	binary edge decision variable such that $x_{i,j} = 1$ if edge (i, j) is included in the solution , 0 otherwise.
y_i	binary relay decision variable such that $y_i = 1$ if a relay is located at node i, 0 otherwise.
$f^k_{i,j}$	binary flow decision variable such that $f^k_{i,j} = 1$ if edge (i, j) is used by commodity k, 0 otherwise.
$u^p_{k,i}$	total distance traveled on a path p by commodity k before node i without visiting a relay.
$v^p_{k,i}$	total distance traveled on a path p by commodity k after node i without visiting a relay.

Formulation 2ECON-NDPR:

$$\text{Min } z = \sum_{i\in V} c^y_i y_i + \sum_{(i,j)\in E, i<j} c^e_{i,j} x_{i,j} \tag{1}$$

$$\sum_{(i,j)\in E} f^k_{i,j} - \sum_{(j,i)\in E} f^k_{j,\,i} = \begin{cases} 2 & i = s(k) \\ -2 & i = t(k) \\ 0 & otherwise \end{cases} \qquad k \in K, i \in V \tag{2}$$

$$u^p_{k,i} \geq v^p_{k,j} + d_{j,i} - (1 - f^k_{j,i})2\lambda \qquad k \in K, i \in V, p \in \{1,2\}, (j,\, i) \in E \tag{3}$$

$$u^p_{k,i} \leq \lambda \qquad k \in K, i \in V, p \in \{1,2\} \tag{4}$$

$$v^p_{k,i} \geq u^p_{k,i} - \lambda y_i \qquad k \in K, i \in V, p \in \{1,2\} \tag{5}$$

$$f^k_{i,j} + f^k_{j,i} \leq x_{i,j} \qquad k \in K, (i, j) \in E, i < j \tag{6}$$

$$y_i, x_{i,j}, f^k_{i,j} \in \{0,1\}$$

$$u^p_{k,i}, v^p_{k,i} \geq 0$$

In the above formulation, Constraint (2) is a set of standard node flow balance constraints to make sure that two unit flows are sent from source node $s(k)$ to destination node $t(k)$. Constraint (3) is used to calculate the total distance traveled by commodity k to node i without visiting a relay on path p. Note that Constraint (3) is valid due to Constraint (6) which limits the flow of the same commodity on each edge to one unit. Constraint (3) becomes $u^p_{k,i} \geq v^p_{k,j} + d_{j,i}$ if commodity k is routed through edge (i, j). If node j is not a relay, then $v^p_{k,j} \geq u^p_{k,j}$ due to Constraint (5), and using this inequality, Constraint (3) can be rewritten as follows:

$$u^p_{k,i} \geq u^p_{k,j} + d_{j,i} \tag{7}$$

On path p, inequality (7) is used to calculate the total distance traveled by commodity k from a relay node to node i without visiting any relay node. This distance is represented by decision variable $u^p_{k,i}$ in the formulation. Constraint (4) is the relay constraint that makes sure that the total distance traveled by commodity k on path p

without visiting a relay node is less than λ at each node. Constraint (5) resets the total distance of path p at a relay node. Note that if node j is a relay, both $v_{k,j}^{p}$ and $u_{k,j}^{p}$ can be zero in Constraint (5). Therefore, inequality (7) and Constraint (3) are reduced to $u_{k,i}^{p} \geq d_{j,i}$ at node i after visiting relay node j. Constraint (6) makes sure that edge (i,j) cannot be used for routing if it is not included in the solution, as well as edge (i,j) cannot be used by each commodity $k$more than once by restricting flows to one unit. Constraints (2) and (6) make sure that there exist 2-edge disjoint paths between every source and destination pair in the network.

Cabral et al. (2007) formulate a path based integer programming model of NDPR and propose a column generation approach which can also be applied in the case of the 2ECON-NDPR. Unfortunately, the column generation approach cannot be practically used to prove the optimality of a solution in the case of large problem instances since all feasible paths and relay locations combinations have to be generated as a priori. Therefore, they recommend using a subset of all feasible paths and relays combinations. Although this approach does not guarantee optimality, superior solutions can be found in reasonable CPU time. In addition, they proposed four different construction heuristics in which a solution is constructed by taking into consideration one commodity at a time. In this paper, we modify three of their construction heuristics to solve the 2ECON-NDPR, which are used as benchmarks to test the performance of our approach. More recently, Kulturel-Konak and Konak (2008) study the NDPR and propose a hybrid local search-genetic algorithm (GA) where a specialized crossover operator based on random paths between source and destination nodes are utilized, and local search procedures are used to investigate new solutions. These local search procedures are not directly applicable to the 2ECON-NDPR. In this paper, we modify this hybrid GA's specialized crossover operator for the 2ECON-NDPR, and the new crossover operator is coupled with customized mutation strategies and a simple heuristic to determine location of relays. Therefore, there is no need for computationally expensive local search procedures.

The NDPR is closely related to a set of well-known network design problems such as the Steiner tree problem, the hop-constrained network design problem, the constrained shortest path problem, and the facility location problem. In the hop constrained network design problem, hop constraints impose an upper bound on the number of edges between some source and some destination nodes due to reliability (LeBlanc and Reddoch 1990) or performance concerns (Balakrishnan and Altinkemer 1992). Several papers (Choplin 2001; Randallet al. 2002; Kwangil and Shayman 2005) address optical network design problems considering restricted transmission range due to optical impairments, which is one of the main motivations in the NDPR.

Similar to the Steiner tree problem, all nodes are not required to be connected in the 2ECON-NDPR. Voss (1999) considers the hop-constrained Steiner tree problem and proposes a solution approach based on tabu search and mathematical programming. Gouveia and Magnanti (2003) consider the diameter-constrained minimum spanning and Steiner tree problems where an upper-bound is imposed on the number of edges between any node pairs. Gouveia (1996) proposes two different mathematical models for the hop-constrained minimum spanning tree problem as well as

Lagrangean relaxation and heuristic approaches to solve the problem. Subsequently, Gouveia and Requejo (2001) develop an improved Lagrangean relaxation approach to the problem. In addition, due to relay decision variables, the 2ECON-NDPR is closely related to the facility location problem (Hakimi 1964).

The objective of this paper is to develop a meta-heuristic approach, namely a genetic algorithm, for the 2ECON-NDPR to solve large problem instances effectively and efficiently. To our best knowledge, 2ECON-NDPR has not previously been studied.

2 A Genetic Algorithm Approach to 2ECON-NDPR

In this section, we describe a genetic algorithm to find good solutions to the 2ECON-NDPR. The parameters and notation used in the GA are given as follows:

z	a solution
$z.x_{i,j}$	edge decision variable of solution z such that $z.x_{i,j} = 1$ if edge (i, j) is selected in solution z, 0 otherwise.
$z.y_i$	relay decision variable of solution z such that $z.y_i = 1$ if a relay is located at node i, 0 otherwise.
$z.p_i(k)$	the i^{th} path of commodity k of solution z.
$E(z)$	edge set of solution $z, E(z) \subset E$
$V(z)$	node set of solution $z, V(z) \subset V$
μ	population size
$t_{\max}$	maximum number of generations (stopping criteria)
P	population
OP	offspring population
$P[i]$	the i^{th} member in the population
U	uniform random number between 0 and 1

2.1 Encoding, Fitness, and Solution Evaluation

By definition, a feasible solution to the 2ECON-NDPR does not have to be a fully connected network. A solution is feasible if at least two edge-disjoint paths exist between source node $s(k)$ and destination node $t(k)$ of each commodity k such that on both paths, the distances between node $s(k)$ and the first relay, between any consecutive relays, and between the last relay and node $t(k)$ and are less than the upper bound λ. This observation is used in the encoding of the GA to represent solutions. Basically, a solution z includes the primary and secondary edge-disjoint paths for each commodity and the relay decision variables, i.e., a solution is represented as $z = \{p_1(1),\ldots,p_1(k); p_2(1),\ldots,p_2(k); y_1,\ldots,y_N\}$. The crossover and mutation operators of the GA generate the edge-disjoint paths $\{p_1(1),\ldots,p_1(k); p_2(1),\ldots,p_2(k)\}$ of an offspring using the edges inherited from two randomly selected parents; and therefore, they ensure two edge-disjoint paths between the source and destination nodes of each commodity. For a given path, relay decision variables $\{y_1, \ldots, y_N\}$ are determined using a simple heuristic during the crossover. This heuristic also ensures feasibility of the paths with respect to the relay constraint. Since the procedures of the GA always generate feasible solutions, the population is ranked according to the cost.

2.2 Specialized Crossover Operator

In a GA, the function of the crossover operator is to create new solutions by recombining the edges of existing solutions. The representation of solutions in the GA is permutation based. When used with the permutation based encodings, traditional GA crossover operators such as single-point or uniform crossover may generate highly infeasible solution structures which have to be repaired by using special repair algorithms. There are a number of crossover operators specially designed for permutation encodings (see (Fogel et al. 1999) for a survey), which can be used to recombine the corresponding paths of parents to create offspring. For example, the corresponding paths of two solutions a and b can be recombined (i.e., crossover $a.p_1(k)$ with $b.p_1(k)$ and $a.p_2(k)$ with $b.p_2(k)$ for each commodity k). There are a few drawbacks associated with such a crossover operator. Firstly, $a.p_1(k)$ and $b.p_1(k)$ may not share any common nodes, which will make the crossover impossible in this case. Secondly, $a.p_1(k)$ and $b.p_2(k)$ or $a.p_2(k)$ and $b.p_1(k)$ may have many overlapping edges. In this case, the crossover of $a.p_1(k)$ with $b.p_1(k)$ must be coordinated with the crossover of $a.p_2(k)$ with $b.p_2(k)$ since the paths of the offspring must be disjoint. Thirdly, the paths of the commodities tend to overlap as the routes of different commodities may share many edges in the 2ECON-NDPR (i.e., $a.p_1(k)$ may share many edges with $a.p_1(j)$ and $a.p_2(j)$ where $k \neq j$). If such dependences between the paths are not considered, an offspring may end up having a much higher number of edges than its parents. Therefore, these dependences between the paths should be considered in order to design an effective crossover for the 2ECON-NDPR.

Because of the reasons briefly discussed above, a specialized crossover is developed based on the Suurballe-Tarjan algorithm (Suurballe and Tarjan 1984). The Suurballe-Tarjan algorithm finds the shortest pair of edge-disjoint paths between a source and a destination node in polynomial time. The crossover operator of the GA is a random construction heuristic in which an offspring solution is constructed from the union of two parent solutions by considering one commodity at a time in a random order of commodities. In Figure 1, a step-by-step example of the crossover and mutation is demonstrated for a problem with two commodities. In the figure, relay nodes are identified by solid circles. The first step in the crossover is randomly selecting two parent solutions, say a and b. After selecting parents, they are combined as illustrated in Figure 1-step (*ii*), and each edge $(i, j) \in E$ of the combined solution is assigned to a temporary cost $tc_{i,j}$ as follows:

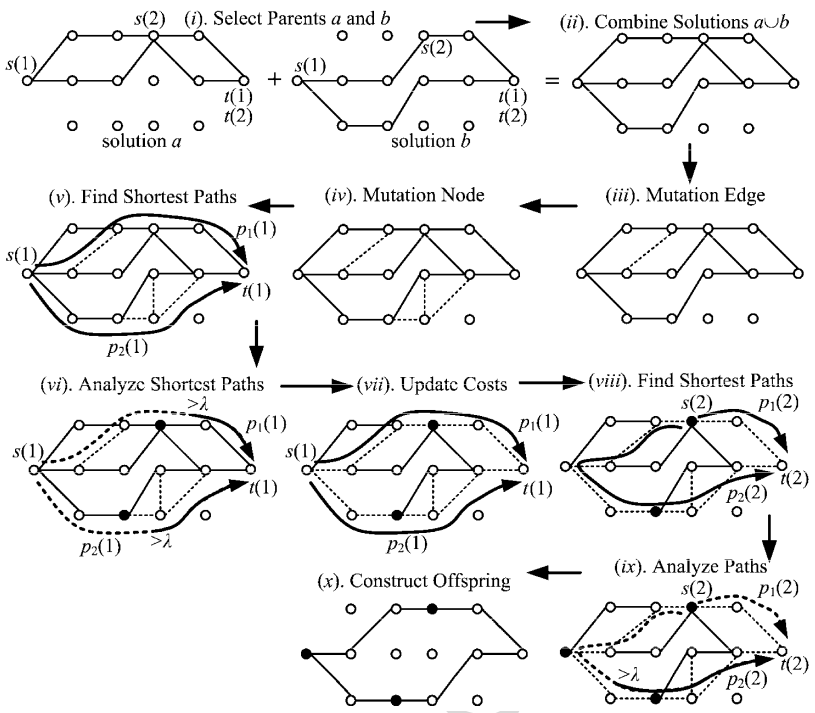

Fig. 1 An example for the crossover and mutation. Dotted edges represent low cost edges.

$$tc_{i,j} = \begin{cases} U \times (c^e_{i,j} + c^v_i + c^v_i) & (i,j) \in E(a) \cup E(b), \\ \infty & \text{otherwise.} \end{cases} \tag{8}$$

Mutation is applied on the combined solution by adding new edges and/or a node. The details of the mutation operators are discussed in the following section. In Figure 1, steps (*iii*) and (*iv*) provide example for edge and node mutations, respectfully. Newly added edges to the combined solution during mutation are assigned random costs (U), which are significantly smaller cost than the ones given in equation (8), so that these new edges are encouraged to be used in the offspring. In Figure 1, such low cost edges are identified by dotted edges.

After mutation, for each commodity k in a random order, two edge-disjoint shortest paths, $p_1(k)$ and $p_2(k)$, from node $s(k)$ to node $t(k)$ are found using the Suurballe-Tarjan algorithm (Suurballe and Tarjan 1984). In Figure 1, step (v) demonstrates the edge-disjoint shortest paths $p_1(1)$ and $p_2(1)$ from node $s(1)$ to node $t(1)$. Note that in this example, commodity 1 is randomly selected as the first commodity to be routed.

After determining paths $p_1(k)$ and $p_2(k)$ of commodity k, the locations of the relays on paths $p_1(k)$ and $p_2(k)$ are determined using a simple heuristic. To do so, starting from node $s(k)$ or $t(k)$ (randomly and uniformly selected), each edge $(i,j) \in p_1(k)$ is sequentially examined whether a relay should be located at node

i or not. After examining edge (i, j), a relay is located at node i if and only if the total distance between node j and the previous relay on path $p_1(k)$ is more than λ. The same procedure is repeated for $p_2(k)$ to determine the locations of relays on this path. For example in step (*vi*), the total distance exceeds λ after traversing the fourth edge on path $p_1(1)$ and the third edge on path $p_2(1)$. Therefore, two relays are allocated on the fourth node of $p_1(1)$ and the third node of $p_2(1)$ for the feasibility of these paths.

Finally, $tc_{i,j}$ is set to zero for all edges $(i, j) \in p_1(k) \cup p_2(k)$ in order to encourage the paths of the unassigned commodities to use already assigned edges. This is illustrated in step (*vii*) where edges of $p_1(1)$ and $p_2(1)$ are dotted, indicating low costs. The example continues with commodity 2. In step (*viii*), two edge-disjoint paths from node $s(2)$ to node $t(2)$ are found. In step (*ix*), these paths are analyzed to locate any necessary relay stations. The resulting offspring is shown in step (*x*). The pseudo code of the crossover operator is presented in the next section.

2.3 Mutation

The purpose of mutation in a GA is to introduce new solution structures (genes) to existing solutions so that local minima can be avoided. In traditional GA mutation, the structure of a solution is randomly changed. A mutation operator of the GA for the 2ECON-NDPR should perturb the edge-disjoint paths by introducing new edges and nodes, which do not exist in parent solutions, while maintaining the edge disjoint paths. The mutation operators of this GA add randomly selected edges or nodes to the parent solutions selected for the crossover. To increase the probability of using these newly added edges and nodes, their costs are set to very small random numbers with respect to existing ones. Then, the crossover operator is applied. By adding new edges and nodes to the parent solutions before applying crossover, the offspring is assured to have two edge-disjoint paths for each commodity. The GA has the following two mutation operators:

Mutation-Edge: Randomly select an edge (i, j) such that $(i, j) \notin \{E(a) \cup E(b)\}, i \in \{V(a) \cup V(b)\}$, and $j \in \{V(a) \cup V(b)\}$ and set $tc_{i,j} := U$. The objective of this mutation is to encourage an offspring to include an edge which does not exist in its parents.

Mutation-Node: Randomly select a node i such that $i \notin \{V(a) \cup V(b)\}$ and there exist at least two edges $(i, j) \in E$ where $j \in \{V(a) \cup V(b)\}$, and then set $tc_{i,j} := U$ for all edges (i, j) where $j \in \{V(a) \cup V(b)\}$. The objective of this mutation is to route commodities through a new node which does not exist in the parents.

The procedure of the mutation and crossover is given below. As mentioned earlier, the mutation operators are applied to the randomly selected parents before applying the crossover. These two mutation operators are sequentially applied independently with the probability of ρ (mutation rate) as given in the procedure below. By setting the cost of new edges introduced by the mutation operators to a random number (U), which is much smaller than the cost of the existing edge on the parents, these new edges are encouraged to be used in the offspring instead of the existing edges of the parents. Mutating the parents before crossover provides significant computational efficiency compared to mutating the offspring by eliminating the need for checking the existence of two edge-disjoint paths between each source and destination pair.

Procedure Crossover_Mutation()

```
Randomly select two solutions $a$ and $b$ from $P$ to create offspring $z$.
**for** each edge $(i, j) \in E$ **do** {
    **if** edge $(i, j) \in E(a) \cup E(b)$ **then** set $tc_{i,j} := U \times (c^e_{i,j} + c^v_i + c^v_i)$
                        **else** set $tc_{i,j} := \infty$}
**if** $(U < \rho)$ **then** apply *Mutation-Edge*
**if** $(U < \rho\ )$ **then** apply *Mutation-Node*
**for** each $k \in K$ in a random sequence **do** {
    find edge disjoint shortest paths $p_1(k)$ and $p_2(k)$ using random costs
    **for** $q := 1, \ldots, 2$ **do** {
        set $Q := 0$
        **for** each edge $(i, j) \in p_q(k)$ **do** {
            set $z.x_{i,j} := 1$
            $Q = Q + d_{i,j}$
            **if** $Q > \lambda$ **then** { set $z.y_i := 1$ and $Q := d_{i,j}$ }
            set $tc_{i,j} := 0$
            }
    }
Return offspring $z$
```

2.4 Overall Algorithm

The important features of the GA are follows:

- The GA is a generational GA where entire population is replaced by offspring generated by the Crossover_Mutation() operator. The GA retains only the best feasible solution found so far in the population between iterations. All other solutions in the next generation are newly generated offspring.
- In the selection procedure, the offspring population is sorted according to the cost. The best μ offspring are selected for the next generation.
- Duplicate solutions are discouraged in the population. If the population has a multiple copies of a solution, only one of them is actually ranked, and the others are placed at the bottom of the population list, and they are not compared with the others. Since comparison of solutions is performed during the ranking procedure, minimum additional computational effort is needed to discover identical solutions in the population. With this approach, the population is discouraged to converge to a single solution, which can be the case in GA. In our initial experiments, we found better results by avoiding identical solutions in the population.

The pseudo code of the GA is given in the following.

Procedure GA

```
    Randomly generate $\mu$ solutions
    for $t := 1, \ldots, t_{\max}$ do {
    //Crossover
    set $i := 0$;
    do {
        $OP[i]$ := Crossover_Mutation();
        Evaluate
        $OP[i]$;
        Update best feasible solution of necessary
        set $i := i + 1$;
    } while ($i < 2\mu$ )
    Rank $OP$ by penalizing duplicate solutions.
    for $j := 1, \ldots, \mu$ do set $P[j] := OP[j]$;
}
```

3 Construction Heuristics

This section presents three construction heuristics based on the ones proposed by Cabral et al. (2007) for the NDPR. As mentioned earlier, in their study, a solution is constructed by considering one commodity at the time using a path based integer programming formulation to solve the NDPR with a single commodity. We used the same approach to construct solutions using the formulation 2ECON-NDPR. The formulation 2ECON-NDPR is a difficult integer programming model to be optimally solved. However, it can be optimally solved in a reasonable time if only a single commodity is considered at a time, which enables us to develop construction heuristics for the 2ECON-NDPR in the similar fashion to those defined in (Cabral et al. 2007) and use the resulting solutions as benchmarks.

Let 2ECON-NDPR(k) represent the formulation of the 2ECON-NDPR with a single commodity k. The formulation 2ECON-NDPR(k) can be used in the three construction heuristics as follows:

Increasing Order Construction Heuristic (IOCH): In this heuristic, the optimal two edge-disjoint paths and relay locations are found for each commodity k by solving the 2ECON-NDPR(k). Let k^* represent the commodity with the minimum cost solution 2ECON-NDPR(k) among all commodities. The paths and relay nodes of the 2ECON-NDPR(k^*) solution are included in the solution, and their associated costs are set to zero. Then, the 2ECON-NDPR(k) is solved again for the remaining commodities using the updated costs, and the paths and relays of the minimum cost solution are added to the solution, and their associated costs are set to zero. The procedure continues in a similar fashion until all commodities are considered. The procedure of IOCH is given below:

Procedure IOCH

```
Set $z.x_{i,j} := 0$ for each $(i,j) \in E$ and $z.y_i := 0$ for each $i \in V$
while $K \neq \emptyset$ {
    Solve the 2ECON-NDPR($k$) for each $k \in K$
    Let $k^*$ be the commodity with the minimum 2ECON-NDPR($k$) solution
    Let $(\mathbf{x}^*, \mathbf{y}^*)$ be decision variables of the 2ECON-NDPR($k^*$)
    for each $(i,j) \in E$ do {
        if $x^*_{i,j} = 1$ then set $z.x_{i,j} := 1$ and $c^e_{i,j} := 0$
    }
    for each $i \in V$ do {
        if $y^*_i = 1$ then $z.y_i := 1$ and $c^v_i := 0$
        $K := K \quad \backslash k^*$
}
Return solution $z$
```

Decreasing Order Construction Heuristic (DOCH): This heuristic is the opposite of the IOCH such that the solution is constructed starting from the maximum cost 2ECON-NDPR(k) solution to the minimum cost one. The procedure of the DOCH is be the same as procedure IOCH with a single difference: k* represents the commodity with the maximum cost 2ECON-NDPR(k) solution among all available commodities.

Random Order Construction Heuristic (ROCH): In this heuristic, a solution is constructed one commodity at a time in a similar fashion to IOCH and DOCH, but in a random order of the commodities. In each iteration a commodity k is selected randomly among available commodities, and the 2ECON-NDPR(k) is solved.

4 Experimental Results & Discussions

In the experimental study, two different problem groups, 80 and 160 nodes are used. For each problem group, the x and y coordinates and cost of each node i are randomly generated from integer numbers between 0 and 100. The cost and distance of each edge (i, j) are defined as the Euclidian distance between nodes i and j. Two different values of λ, 30 and 35, are tested. Edges longer than λ are not considered in the problems. Therefore, the number of edges (M) for each problem depends on λ. For example, the 160 node problem with $\lambda = 35$ has 3,624 edges and the ones with $\lambda = 30$ has 2,773 edges. In addition, three different levels of $K, 5, 10$ and 15, are considered. The source and the destination nodes of commodities for each problem are randomly selected but forced to be far apart from each other so that the problems are not trivial to solve. In Table 1, the problems are named using their parameters (e.g., the problem with 160 nodes, 10 commodities and $\lambda = 35$ is named as 160-10-35).

Table 1 Results of the Test Problems

				GA	Avg.	IP			
Problem		GA	GA	% Coef.	CPU	Model			
($N-K-\lambda$)	M	Best	Avg.	Var	Sec*	(% Opt. Gap)	IOCH	DOCH	ROCH
80-5-30	641	616.2	623.5	0.39	285	662.8 (32%)	758.9	773.0	746.7
80-5-35	853	527.7	527.8	0.02	309	593.7 (29%)	701.8	722.0	718.4
80-10-30	641	708.3	715.2	0.50	363	757.5 (35%)	1,025.5	1070.3	973.7
80-10-35	853	628.3	647.7	0.81	369	662.7 (31%)	1,004.9	1,018.8	922.8
80-15-30	641	880.5	914.3	2.16	432	934.8 (42%)	1,265.0	1,281.2	1,151.2
80-15-35	853	805.8	853.5	2.59	391	879.2 (41%)	1,236.7	1,169.7	1,126.0
160-5-30	2,773	420.6	453.7	5.17	348	545.1 (38%)	598.5	654.5	590.1
160-5-35	3,624	421.0	443.8	3.67	366	496.8 (33%)	568.8	625.5	599.6
160-10-30	2,773	590.9	617.6	2.96	494	769.5 (46%)	879.8	1,018.9	932.2
160-10-35	3,624	560.9	596.3	3.40	509	651.0 (36%)	913.6	926.0	861.7
160-15-30	2,773	705.5	745.0	4.60	642	876.4(45%)	1,128.4	1,159.3	1,077.7
160-15-35	3,624	696.9	739.9	3.41	688	835.8 (43%)	1,155.4	1126.8	1,075.2

*The computational experiments were performed on a PC with 2.66GHZ Intel Dual-Core CPU and 4GB memory.

Table 1 presents the results found by the GA, the construction heuristics, and the integer programming. To gauge GA's results, formulation 2ECON-NDPR was solved using CPLEX V9.0 with a time limit of 24 CPU hours. Unfortunately, none of the problems could be solved to optimality within the time limit. Therefore, the best feasible solutions found and their percent optimality gaps at the termination are reported in Table 1. In the construction heuristics, each sub-problem 2ECON-NDPR(k) was solved using CPLEX V9.0 with a time limit of 1,800 CPU seconds. If the optimal solution of 2ECON-NDPR(k) could not be obtained within this time limit, the best feasible solution was used in the construction heuristics.

The GA was run for 30 random replications with parameters $\mu = 50$, $\rho = 0.1$, and $t_{\max} = 1000$. The best feasible solution, the average, and the percent coefficient of variation of random 30 runs are provided in the table. In each case, the GA was able to find a significantly better solution than the best solution found by integer programming. As it can be seen in the table, the GA outperforms the construction heuristics in all problems. The GA and the construction heuristics are similar in the way solutions are created one commodity at a time. In the construction heuristics, the solution for a single commodity is optimal for that commodity and they are therefore myopic. In the GA, on the other hand, the solution for a single commodity is randomly generated using random costs in the crossover and mutation. Therefore, the GA may consider a broader range of solutions than the greedy construction heuristics.

The GA is highly robust over random replications. The percent coefficient of variation is less than 5% for all cases. As seen in the CPU times, the GA scales well. Computationally, the most expensive operation of the GA is the Suurballe-Tarjan algorithm to find the shortest pairs of edge-disjoint paths, which can be done in $O(M \log_{1+M/N} N)$ (Suurballe and Tarjan 1984). Since the Suurballe-Tarjan algorithm runs for each commodity, CPU time depends on the number of the commodities routed in the network. However, we observed that after finding the shortest

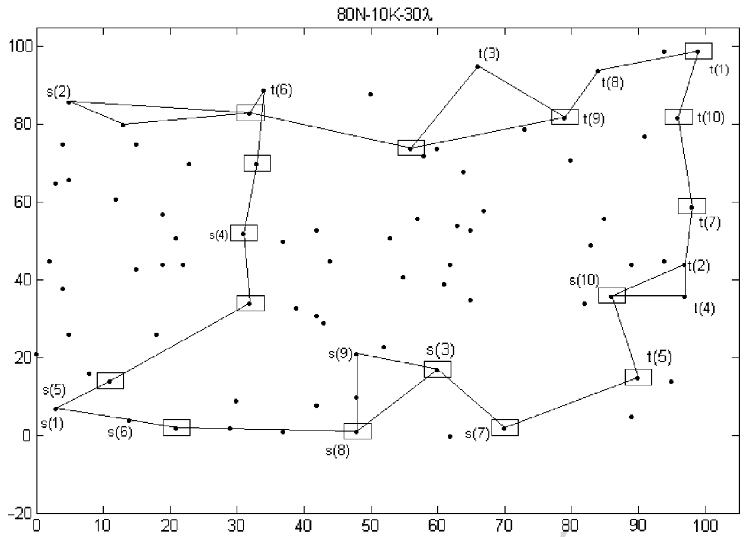

Fig. 2 The best GA solution found for 80-10-30.

paths for the first few commodities, the shortest paths for the remaining ones are determined much faster since they tend to overlap with the already added paths. Therefore, the CPU times did not linearly increase with the number of commodities as it might be expected. Overall, the GA is capable of solving large size problems in reasonably short times. Figure 2 illustrates the best solution found for problem 80-10-30 by the GA.

5 Conclusions and Future Research

In this paper, a GA with specialized crossover and mutation operators was developed to solve the survivable network design problem with relays for the first time. Our initial experiments demonstrate that the GA is very promising in both computational efficiency and in optimization performance. The proposed crossover approach can be extended to other survivable network design problems such as node-disjoint network design problem as a further research.

References

Balakrishnan, A, Altinkemer, K (1992) Using a hop-constrained model to generate alternative communication network design. ORSA Journal on Computing 4: 192-205

Cabral, EA, Erkut, E, Laporte, G, et al. (2007) The network design problem with relays. European Journal of Operational Research 180: 834-844

Choplin, S (2001). Virtual path layout in ATM path with given hop count. Proceedings of the First International Conference on Networking-Part 2, Colmar, France, Springer-Verlag.

Fogel, DB, Bäck, T, Michalewicz, Z (1999) Evolutionary Computation 1. IOP Publishing Ltd, Bristol, UK

Gouveia, L (1996) Multicommodity flow models for spanning trees with hop constraints. European Journal of Operational Research 95: 178-190

Gouveia, L, Magnanti, TL (2003) Network flow models for designing diameter-constrained minimum spanning and Steiner trees. Networks 41: 159-173

Gouveia, L, Requejo, C (2001) A new Lagrangean relaxation approach for the hop-constrained minimum spanning tree problem. European Journal of Operational Research 132: 539-552

Hakimi, SL (1964) Optimum locations of switching centers and absolute centers and medians of graph. Operations Research 12: 450-459

Kulturel-Konak, S, Konak, A (2008) A Local Search Hybrid Genetic Algorithm Approach to the Network Design Problem with Relay Stations. in S. Raghavan, B. L. Golden and E. Wasil (ed) Telecommunications Modeling, Policy, and Technology. Springer, New York

Kwangil, L, Shayman, MA (2005) Optical network design with optical constraints in IP/WDM networks. IEICE Transactions on Communications E88-B: 1898-1905

LeBlanc, L, Reddoch, R (1990). Reliable link topology/capacity design and routing in backbone telecommunication networks. First ORSA Telecommunications SIG Conference.

Magnanti, TL, Mirchandani, P, Vachani, R (1995) Modeling and solving the two-facility capacitated network loading problem. Operations Research 43: 142-157

Monma, CL, Shallcross, DF (1989) Methods for designing communications networks with certain two-connected survivability constraints. Operations Research 37: 531-541

Randall, M, McMahon, G, Sugden, S (2002) A simulated annealing approach to communication network design. Journal of Combinatorial Optimization 6: 55-65

Suurballe, JW, Tarjan, RE (1984) A quick method for finding shortest pairs of disjoint paths. Networks 14: 325-336

Voss, S (1999) The Steiner tree problem with hop constraints. Annals of Operations Research 86: 321-345