Chapter 15

A LOCAL SEARCH HYBRID GENETIC ALGORITHM APPROACH TO THE NETWORK DESIGN PROBLEM WITH RELAY STATIONS

Sadan Kulturel-Konak, Abdullah Konak
Penn State Berks, Tulpehocken Road, P.O. Box 7009, Reading, PA 19610, US

Abstract: The network design problem with relay stations arises in telecommunication and logistic systems. The design problem involves selecting network links and determining the location of the relay stations to minimize the design cost. A constraint is imposed on the distance that a signal can travel without being regenerated by a relay station. An efficient hybrid meta-heuristic approach is presented to solve large sized problems.

Keywords: Network design; telecommunications; genetic algorithms; meta-heuristics.

1. INTRODUCTION

The network design problem with relay stations (NDPRS) is briefly defined as follows. A 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) has an installation cost of $c_{i,j}$ and a length 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 $t(k)$. All traffic from node $s(k)$ to node $t(k)$ is routed through a single path. Being different from a regular network design problem, some nodes of the network have to be dedicated as relay stations in which the communication signal is regenerated. An upper bound λ is imposed on the distance that a commodity k can travel without visiting a relay station on a path from node $s(k)$ to node $t(k)$. A fixed cost f_i is occurred when a relay station is located at node i. The objective function of the NDPRS is to minimize the network design cost while making sure that each traffic commodity k is routed from node $s(k)$ to node $t(k)$ through a path on which the distances between node $s(k)$ and the

first relay station, between any two consecutive relay stations, and between node $t(k)$ and the last relay station are less than the upper bound λ. A network flow based formulation of the NDPRS 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 selected in the solution, 0 otherwise.
y_i	binary relay station variable such that $y_i=1$ if a relay station is located at node i, 0 otherwise.
$\overline{x}_{i,j}^k$	binary flow decision variable such that $\overline{x}_{i,j}^k=1$ if edge (i, j) is used by commodity k, 0 otherwise.
$u_{k,i}$	total distance traveled by commodity k to node i without visiting a relay station.
$u'_{k,i}$	total distance traveled by commodity k after node i without visiting a relay station.

Problem NDPRS:

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

$$\sum_{(i,j) \in E} \overline{x}_{i,j}^k - \sum_{(j,i) \in E} \overline{x}_{j,i}^k = \begin{cases} 1 & i = s(k) \\ -1 & i = t(k) \\ 0 & otherwise \end{cases} \qquad k \in K, i \in V \tag{2}$$

$$u_{k,i} \geq u'_{k,j} + d_{j,i} - (1 - \overline{x}_{j,i}^k)M \qquad k \in K, i \in V, (j,i) \in E \tag{3}$$

$$u_{k,i} \leq \lambda \qquad k \in K, i \in V \tag{4}$$

$$u'_{k,i} \geq u_{k,i} - My_i \qquad k \in K, i \in V \tag{5}$$

$$\sum_{k \in K} (\overline{x}_{i,j}^k + \overline{x}_{j,i}^k) \leq Mx_{i,j} \qquad (i,j) \in E, i < j \tag{6}$$

$$y_i, x_{i,j}, \overline{x}_{i,j}^k \in \{0,1\}$$

$$u_{k,i}, u'_{k,i} \geq 0$$

In the formulation above, constraint set (2) is the node flow balance constraints to make sure that one unit flow is sent from node $s(k)$ to node $t(k)$. Constraint (3) is used to calculate the total distance traveled by each commodity without visiting a relay station. Constraints (4) is the relay station constraint that makes sure that the total distance traveled by any commodity without visiting a relay station is less than λ at each node.

Constraint (5) resets the total distance traveled by commodity k until node i to zero if node i is a relay station. Constraint (6) makes sure that edge (i, j) cannot be used by any commodity if it is not included in the solution.

The NDPRS is first introduced by Cabral *et al.* [1] in the context of a real life telecommunication network design problem. Cabral *et al.* [1] formulate a path based integer programming model of the problem and propose a column generation approach. Unfortunately, the column generation approach cannot be practically used to find the optimal solution in the case of large problem instances since all feasible paths and relay station locations combinations have to be generated as a priori. Therefore, Cabral *et al.* [1] recommend using a subset of all feasible paths and relay station combinations. Although this approach does not guarantee optimality, superior solutions can be found in reasonable CPU times. In addition, they proposed four different construction heuristics in which a solution is constructed by taking into consideration one commodity at a time.

Although the NDPRS was studied first time by Cabral *et al.* [1], it is closely related to 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 source and destination node pairs due to reliability [2] or performance concerns [3]. Several papers [4-6] address optical network design problem considering restricted transmission range due to optical impairments, which is one of the main motivations in the NDPRS as well.

Similar to the Steiner tree problem, all nodes are not required to be connected in the NDPRS. Voss [7] considers hop-constrained Steiner tree problem and proposes a solution approach based on tabu search and mathematical programming. Gouveia and Magnanti [8] 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 [9] provides two different mathematical models for the hop-constrained minimum spanning tree problem; and, in addition, he proposes Lagrangean relaxation and heuristic approaches to solve the problem. Later, Gouveia and Requejo [10] develop an improved Lagrangean relaxation approach to the problem. In a sense, the NDPRS is a generalization of the hop-constrained network design problem where $d_{i,j}$=1 for each edge and λ is an integer. In addition, due to relay station decision variables, the NDPRS is closely related to the facility location problem [11].

The objective of this paper is to develop a meta-heuristic approach for the NDPRS to solve large problem instances effectively and efficiently. After experimenting with a few different meta-heuristics approaches such as

Genetic Algorithms (GA), Simulated Annealing, and Ant Colony Approach, the best results were found by a local search hybrid genetic algorithm. It should be noted that no meta-heuristic approach has been previously applied to this problem. We report the results found by a local search hybrid genetic algorithm.

2. A HYBRID GENETIC ALGORITHM APPROACH FOR THE NETWORK DESIGN WITH RELAY STATIONS

In this section, we describe a hybrid genetic algorithm to find good solutions to the NDPRS. The algorithm is called local search hybrid genetic algorithm (LSHGA) since a local search is used to explore new solutions in addition to the traditional GA operators, crossover and mutation. The parameters and notation used in the LSHGA 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 station is located at node i, 0 otherwise.
$E(z)$	edge set of solution z, $E(z) \subset E$
$V(z)$	node set of solution z, $V(z) \subset V$
μ	population size
$tc_{i,j}$	temporary cost of edge (i, j)
$t_{\max}$	maximum number of generations (stopping criteria)
P	population
OP	offspring population
$P[i]$	the i^{th} solution in the population
U	uniform random number between 0 and 1
N	number of nodes, $\|V\|$
M	number of edges, $\|E\|$

2.1 Encoding, Fitness, and Solution Evaluation

A binary encoding based on the edge and relay decision variables of the NDPRS is used to represent a solution z as described above. By definition, a feasible solution to the NDPRS does not have to be a connected network. A solution is feasible if at least one path exists between the source and destination nodes of each commodity and the commodities are routed no longer than λ distance without visiting a relay station. The

crossover/mutation operators and local search procedures of the LSHGA ensure that the source and destination nodes of each commodity are always connected. However, the feasibility of solutions with respect to the relay station distance constraint is not guaranteed. Therefore, the feasibility of solutions is checked using a depth-first search. The number of commodities that violate the relay station distance constraint of the problem NDPRS is used as a measure of infeasibility. Infeasible solutions are not discarded from the population, but penalized using a ranking schema as follows: (*i*) between two feasible solutions, the one with the lower cost is superior; (*ii*) between an infeasible and a feasible solution, the feasible solution is superior; (*iii*) between two infeasible solutions, the one with a smaller constraint violation is superior. Instead of assigning a fitness value, the population is ranked according to these three rules. This ranking procedure is simple and does not require any parameters unlike penalty function approaches.

2.2 Crossover and Mutation

In GA, the function of crossover is to create new solutions by recombining the characteristics of existing solutions. In our initial experiments with GA, traditional GA crossover operators such as single-point or uniform crossover did not provide promising results since they generated highly disconnected and infeasible solutions. Therefore, a specialized crossover based on random shortest paths between source and destination nodes was developed. An offspring solution z is constructed from two parent solutions $P[p_1]$ and $P[p_2]$ by considering one commodity at a time in a random order of commodities. After randomly selecting indexes p_1 and p_2 for parent solutions, temporary cost $tc_{i,j}$ is assigned to each edge as follows: for each edge $(i, j)\in E$, if $P[p_1].x_{i,j}=1$ or $P[p_2].x_{i,j}=1$, then set $tc_{i,j}:=U\times(c_{i,j}+f_i+f_j)$; else, set $tc_{i,j}:=\infty$. Then, the shortest path $p(k)$ from node $s(k)$ to node $t(k)$ with respect to temporary costs is found for each commodity k in a random order of commodities. After finding the shortest path $p(k)$, each edge $(i, j)\in p(k)$ is added to the offspring while setting its temporary cost $tc_{i,j}$ to zero in order to encourage unassigned commodities to use the edges already included in the offspring. This process is repeated until all commodities are routed. Finally, the locations of the relay stations on path $p(k)$ are determined based on the relay stations of parents $P[p_1]$ and $P[p_2]$. To do so, starting from node $s(k)$, each edge $(i, j)\in p(k)$ is sequentially examined whether a relay station should be located at node i or not. After examining edge (i, j), a relay station is located at node i if and only if the total distance from the previous relay station on path $p(k)$ to node j is more than λ and either of the parents has a relay station located at node i.

The LSHGA has also a specialized mutation operator. The function of mutation in GA is to avoid local minima by preventing solutions to become too similar to each other. The mutation operator of the LSHGA randomly modifies a solution while making sure that a path exists between the source and destination nodes of commodities. To achieve this, mutation is applied within the crossover operator. First, an edge (i, j) such that $(i, j) \notin E(P[p_1])$, $(i, j) \notin E(P[p_2])$, $i \in \{V(P[p_1]) \cup V(P[p_2])\}$, and $j \in \{V(P[p_1]) \cup V(P[p_2])\}$ is randomly selected, and then its temporary cost $tc_{i,j}$ is set to zero before applying the crossover. By randomly selecting an edge that does not exist in either of the parents and setting its cost to zero, it is expected that the offspring will include a different edge than its parents. Mutation is applied to ρ percent of the population. The procedure of the crossover/mutation operator is given as follows:

Procedure Crossover_Mutation()

Randomly select two solutions $P[p_1]$ and $P[p_2]$ from P.
for each edge $(i, j) \in E$ **do** {
    if $P[p_1].x_{i,j} = 1$ or $P[p_2].x_{i,j} = 1$ **then** $tc_{i,j} := U \times (c_{i,j} + f_i + f_j)$ **else** $tc_{i,j} := \infty$
    }
if $(\rho < U)$ **then** {
    randomly select an edge (i, j) such that $(i, j) \notin E(P[p_1])$,
    $(i, j) \notin E(P[p_2])$, $i \in \{V(P[p_1]) \cup V(P[p_2])\}$, & $j \in \{V(P[p_1]) \cup V(P[p_2])\}$.
    set $tc_{i,j} := 0$
    }
for each commodity k in a random sequence of commodities **do** {
    find the shortest path $p(k)$ from node $s(k)$ to node $t(k)$ using
    temporary costs $tc_{i,j}$.
    set $Q := 0$
    for each edge $(i, j) \in p(k)$ **do** {
        set $z.x_{i,j} := 1$
        $Q := Q + d_{i,j}$
        if $Q > \lambda$ and ($P[p_1].y_i = 1$ or $P[p_2].y_i = 1$) **then** {set $z.y_i := 1$ and
                                                                $Q := d_{i,j}$}
        set $tc_{i,j} := 0$
    }
}

2.3 Local Search

As mentioned earlier, the LSHGA also uses local search to investigate new solutions in addition to crossover and mutation. In the local search, a set of neighborhood solutions of each solution in the population is searched by applying a graph perturbation operator, and the best solution in the neighborhood, which is determined by the population ranking rules given in Section 2.1, is selected as the offspring. It should be noted that the best solution in a neighborhood might be worse than the original solution. Therefore, the local search also prevents getting mired at local optima. We considered several local search operators and after initial experiments four different types were selected to be used in the LSHGA. These four different local search procedures are described as follows:

Node Swap Local Search: This local search is based on a node swap operator which replaces a node that is in the solution with another node that is not. For a solution z, swapping nodes i and j is said to be admissible if $i \in V(z)$, $j \notin V(z)$, and edge $(j, k) \in E$ for each edge $(i, k) \in E(z)$. The node swap operator for an admissible node pair i and j is given as follows:

node_swap(i, j)
for each edge $(i, k) \in E(z)$ **do** {
    set $z.x_{j,k}$:=1 and $z.x_{i,k}$:=0
    }
set $z.y_j$:= $z.y_i$

For a solution z, all admissible node_swap(i, j)s are performed, and the best solution is added to the offspring population.

Relay Station Local Search: In this local search, the values of the relay station decision variables are flipped (i.e., $z.y_i$=1−$z.y_i$) one node at a time for each node $i \in V(z)$. As a result, the best solution is returned.

Edge Swap/Node Add Local Search: In this local search, a solution z is modified by removing an edge $(i, j) \in E(z)$ and adding two new edges (i, k) and (j, k). Node k is included in the solution if it does not exist in the solution. This perturbation is permissible only if $(i, k) \in E$ and $(j, k) \in E$. As a result of this perturbation, nodes i and j are connected through node k instead of being directly connected; therefore, the connectivity of solution z is not disturbed. Similarly, after performing all permissible perturbations, the best solution is added to the offspring population.

Edge Swap/Node Delete Local Search: This local search is the opposite of the edge swap/node add local search. Basically, two edges $(i, j) \in E(z)$ and $(j, k) \in E(z)$ are removed from a solution z, and new edge $(i, k) \notin E(z)$ is added (i.e, set $z.x_{i,j}$:=0, $z.x_{i,k}$:=0, and $z.x_{i,j}$:=1). Deletion of edges (i, j) and (j, k) is permissible if

– $(i, k) \in E$ and $(i, k) \notin E(z)$, and
– node j has a node degree of two, and
– node j is neither a source node nor a destination node.

The local search is applied to all permissible node triples i, j, and k, and the best solution is added to the offspring population.

2.4 Overall Algorithm

The important features of the LSHGA are given as follows:

– Initial solutions are randomly generated in a similar fashion to the crossover operator on the complete edge and node sets. A uniform random number is assigned to each edge ($tc_{i,j}$:=U for each $(i, j) \in E$), and the shortest path from node $s(k)$ and node t(k) is found for each commodity k in a random order of commodities. After finding the shortest path from node $s(k)$ to node $t(k)$, the nodes on the shortest path is sequentially analyzed starting from node $s(k)$ in order to determine the location of relay stations on the shortest path. The edges on the shortest path and relay nodes are added to the solution, and $tc_{i,j}$:=0 for each edge included in the solution.
– The LSHGA is a generational GA where the entire parent population is replaced by offspring generated by either the crossover/mutation or the local search. In initial experiments, significantly better solutions were obtained with a generational GA than with a steady-state GA where the population retains the best solutions of earlier generations. The LSHGA retains only the best feasible solution found so far in the population between iterations. All other solutions in the next generation are newly generated.
– Duplicate solutions are discouraged in the population. If a solution happens to have multiple copies in the offspring population, only one of them is considered while ranking offspring solutions, and the others are automatically assigned a low rank (i.e., they are pushed back in the offspring population). Thereby, the population is discouraged from converging to a single solution, which is usually the case in GA. In our initial experiments, however, we found better results by avoiding identical solutions in the population.
– In the local search, one of the four local search procedures given in Section 2.3 is randomly and uniformly selected.
– In the selection procedure, the offspring population is sorted according to the solution ranking rules given in Section 2.1. The best μ-1 offspring are selected for the next generation.

The pseudo code of the LSHGA is given as follows:

Procedure LSHGA

Randomly generate μ solutions
for t:=1..t_{max} **do** {
  //Crossover
  i:=0
  do {
    Randomly select two distinct integers p_1 and p_2 in [1, μ]
    $OP[i]$:=Crossover_Mutation($P[p_1]$, $P[p_2]$)
    Evaluate $OP[i]$
    i:=i+1
  } **while** ($i < \mu$)
// Local Search
  for j:=1..μ **do** {
    Randomly and uniformly select a local search procedure
    Apply local search, set $OP[i]$:=Local_Search($P[j]$)
    i:=i+1
    }
  Rank and sort OP according to the ranking rules (consider a single copy of identical solutions while ranking OP and assign a low ranking to the others)
  for j:=1..μ **do** set $P[j]$:=$OP[j]$
}

3. EXPERIMENTAL RESULTS

In the experimental study, 20 problem instances of five groups of different number of nodes, the smallest with 40 nodes and 198 edges and the largest with 160 nodes and 3624 edges, are used. For each problem group, the x and y coordinates and relay station cost f_i of each node i are randomly generated from integer numbers between 0 and 100. Cost $c_{i,j}$ and distance $d_{i,j}$ of edge (i, j) are defined as the Euclidian distance between nodes i and j. Two different values of λ and K are used. The source and the destination nodes of each commodity for each problem group are randomly selected while making sure that the problems are not trivial (source and destinations nodes are selected apart from each other). The settings for the problems are given in Table 1. In this section, we refer to individual problems using problem parameters N-K-λ. For example, 160N-10K-35λ refers to the problem with 160 nodes, 10 commodities, and λ=35.

We compared the performance of the LSHGA with the Construction Heuristic 1 (CH1) and Construction Heuristic 2 (CH2) developed by Cabral *et al*. [1]. In the CH1, a solution is constructed by considering one

commodity at the time in a random order of commodities. Cabral *et al.* [1] used a path based integer programming formulation to solve the NDPRS with a single commodity. The NDPRS with a single commodity is called the subproblem of in the CH1. This formulation approach requires that all feasible path and relay station location combinations between a source node and a destination node are generated a priori and input to the model. Unfortunately, generating all feasible path and relay station location combinations is computationally infeasible for the size of problems studied in this paper. In such cases, Cabral *et al.* [1] recommend using a subset of all feasible path and relay station location combinations to find good solutions to the subproblems without guaranteeing optimality. Fortunately, we were able to optimally solve the problem NDPRS with a single commodity for all problem instances using CPLEX 9.0. It should be noted that since the subproblems of the CH1 are solved to the optimality in this paper, the CH1 run longer than the CPU times reported in [1].

The CH2 provides the best results among the four construction heuristics given in [1]. The CH2 uses the CH1 as a subroutine and analyzes each edge and node one at the time to test whether an edge or node should be included in the final solutions or not. Therefore, the CH2 is a computationally very expensive heuristic. In order to solve problem 160*N*-10*K*-35λ, for example, the CH2 requires minimum of 10×(160+3624) calls to the CH1 (at least 10 calls for each node and edge). Cabral *et al.* [1] recommend using the CH1 if the CPU time is premium. Due to the time constraints, we were able to implement CH2 for all instances of the 40, 50, 60-node problems and only one instance of the 80-node problem, but not for 160-node problems.

The total costs for the best solutions found by the LSHGA, CH1, and CH2 are given in Table 1. The parameters of the LSHGA were μ=50, ρ=0.50, and t_{max}=100 in all runs. For each problem setting, 10 random replications were performed. The average cost of 10 random replications for each problem is also given in the table. To be consistent, the CH1 was run for 10 random replications and the best solution found in 10 replications is reported in Table 1. As seen in the table, the best solutions found by the LSHGA are superior to those found by the CH1 and CH2, excluding problem 80*N*-5*K*-30λ. For this problem, the CH2 found a better solution. In some cases, even the average costs of 10 random replications are better than the best objective function values found by the CH1 and CH2. In every problem, the average cost of 10 random replications is very close to the best objective function value found. This shows the robustness of the LSHGA over random replications.

In Table 1, the CPU times given for the LSHGA are averaged over 10 replications. The CPU time results show that the LSHGA is highly scaleable. The CPU time requirement was only doubled from the smallest

(with 198 edges) to the largest problem (with 3624 edges) while the problem size increased in 18-fold. The CPU time requirement for the CH2 was in the range of several hours since subproblems were solved to optimality using the problem NDPRS.

Table 1. Results of the test problems

				LSHGA (10 Replications)			CH1	CH2
N	*K*	λ	*M*	Best Cost	Average Cost	Average CPU Seconds*.	Best Cost	Best Cost
40	5	30	198	473.80	475.57	391	486.55	486.55
40	5	35	272	354.57	356.56	457	361.49	381.27
40	10	30	198	518.98	519.16	443	567.76	547.20
40	10	35	272	399.76	409.16	491	446.45	434.56
50	5	30	279	283.78	283.78	392	341.34	320.31
50	5	35	372	260.23	260.23	383	260.23	267.31
50	10	30	279	540.39	550.03	581	592.71	594.29
50	10	35	372	407.49	418.72	688	467.50	481.46
60	5	30	305	509.90	524.02	433	525.41	538.38
60	5	35	412	377.02	377.14	481	404.51	397.44
60	10	30	305	694.89	732.11	642	714.65	731.41
60	10	35	412	499.64	512.88	645	595.86	596.16
80	5	30	641	356.65	360.70	412	372.29	341.72
80	5	35	853	328.80	332.18	455	347.05	
80	10	30	641	464.99	482.81	514	501.45	
80	10	35	853	436.75	459.84	539	516.90	
160	5	30	2773	287.84	292.27	538	297.25	
160	5	35	3624	270.22	274.27	569	314.51	
160	10	30	2773	408.10	431.60	643	458.37	
160	10	35	3624	409.93	433.29	775	479.87	

*The computational experiments were performed in the Lion-XM system of the High Performance Computing Group at the Pennsylvania State University (http://gears.aset.psu.edu/hpc/systems/lionxm/). The implementation code was not paralyzed over multiple processors. Therefore, the CPU times are comparable with a single 3.2 GHz Intel Xeon Processor with 4GB memory.

Figures 1 and 2 illustrate the best solutions found for problems 80*N*-10*K*-30λ and 80*N*-10*K*-35λ by the LSHGA, respectively. On both figures, the source and destination nodes are also identified. In both cases, the best solutions are spanning trees.

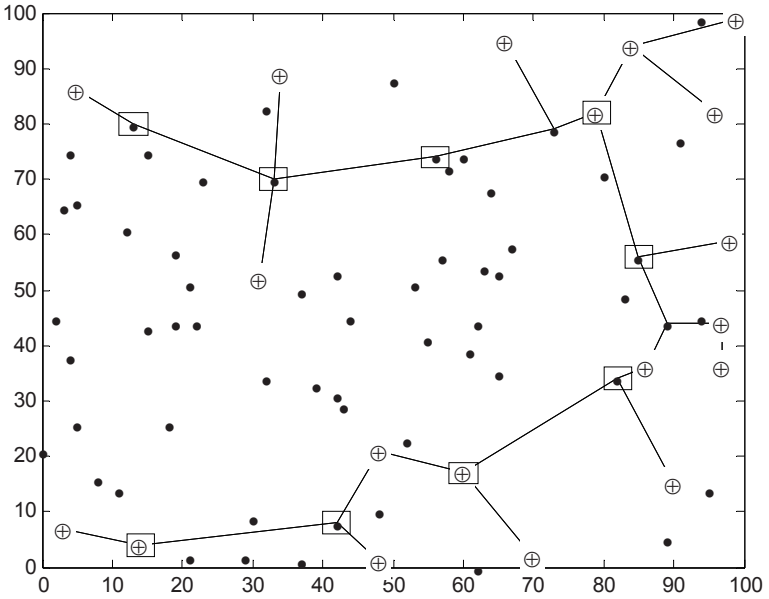

Figure 1. The best solution found for problem 80*N*-10*K*-30λ (●: location of nodes, □: relay station node, and ⊕: source or destination node)

4. CONCLUSIONS AND FUTURE RESEARCH

In this paper, to our best knowledge, a meta-heuristic based approach was applied to the network design problem with relay stations for the first time. Our initial experiments with traditional GA and local search approaches did not provide promising results. Therefore, a hybrid approach was developed. The proposed hybrid GA approach has specialized crossover and mutation operators, and four different local search procedures are used. The experimental results showed that the proposed hybrid GA with the local search heuristics was effective to find good solutions in reasonable CPU times. Compared to the CH1 and CH2 [1], the proposed approach was superior.

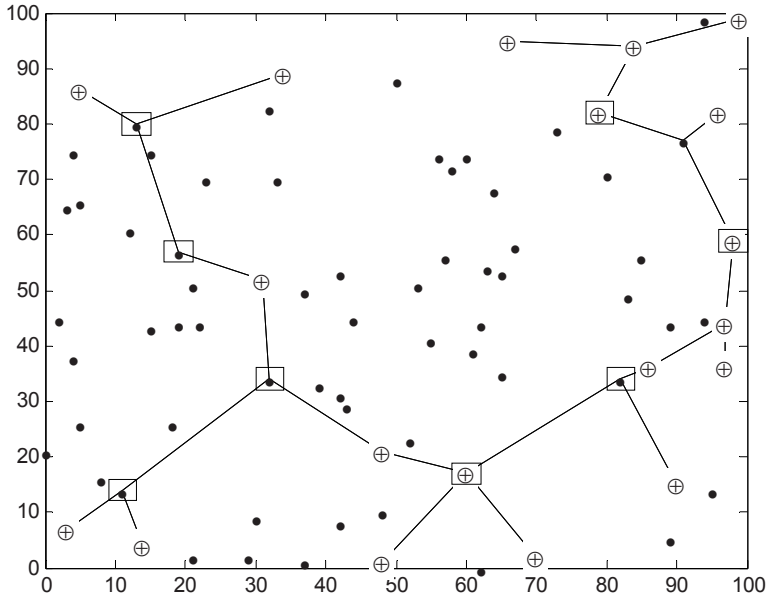

Figure 2. The best solution found for problem 80*N*-10*K*-35λ (●: location of nodes, □: relay station node, and ⊕: source or destination node)

References:

1. E. A. Cabral, E. Erkut, G. Laporte, and R. A. Patterson, The network design problem with relays. *European Journal of Operational Research* 180(2), 834-844, (2007).
2. L. LeBlanc and R. Reddoch, Reliable link topology/capacity design and routing in backbone telecommunication networks. *First ORSA Telecommunications SIG Conference*, (1990).
3. A. Balakrishnan and K. Altinkemer, Using a hop-constrained model to generate alternative communication network design. *ORSA Journal on Computing* 4(2), 192-205, (1992).
4. S. Choplin, Virtual path layout in ATM path with given hop count. *Proceedings of the First International Conference on Networking-Part 2*, Colmar, France, 527-537, (2001).
5. L. Kwangil and M. A. Shayman, Optical network design with optical constraints in IP/WDM networks. *IEICE Transactions on Communications* E88-B(5), 1898-905, (2005).
6. M. Randall, G. McMahon, and S. Sugden, A simulated annealing approach to communication network design. *Journal of Combinatorial Optimization* 6(1), 55-65, (2002).
7. S. Voss, The Steiner tree problem with hop constraints. *Annals of Operations Research* 86(0), 321-345, (1999).
8. L. Gouveia and T. L. Magnanti, Network flow models for designing diameter-constrained minimum spanning and Steiner trees. *Networks* 41(3), 159-73, (2003).

9. L. Gouveia, Multicommodity flow models for spanning trees with hop constraints. *European Journal of Operational Research* 95(1), 178-90, (1996).
10. L. Gouveia and C. Requejo, A new Lagrangean relaxation approach for the hop-constrained minimum spanning tree problem. *European Journal of Operational Research* 132(3), 539-52, (2001).
11. S. L. Hakimi, Optimum locations of switching centers and absolute centers and medians of graph. *Operations Research* 12(3), 450-459, (1964).