Genetic Algorithm for Network Cost Minimization Using Threshold Based Discounting

HRVOJE PODNAR  
opodnar@southernct.edu
Computer Science Department, Southern Connecticut State University, New Haven, CT 06515, USA

JADRANKA SKORIN-KAPOV†  
jskorin@notes.cc.sunysb.edu
W.A. Harriman School for Management and Policy, State University of New York at Stony Brook, Stony Brook, NY 11794-3775, USA

Abstract. We present a genetic algorithm for heuristically solving a cost minimization problem applied to communication networks with threshold based discounting. The network model assumes that every two nodes can communicate and offers incentives to combine flow from different sources. Namely, there is a prescribed threshold on every link, and if the total flow on a link is greater than the threshold, the cost of this flow is discounted by a factor $\alpha$. A heuristic algorithm based on genetic strategy is developed and applied to a benchmark set of problems. The results are compared with former branch and bound results using the CPLEX® solver. For larger data instances we were able to obtain improved solutions using less CPU time, confirming the effectiveness of our heuristic approach.

Keywords: genetic algorithm, mixed integer programming, threshold based discounting, network design

1. Introduction

In this paper we address communication networks that support user’s cooperation in utilizing network links. This is a continuation of work addressed in Podnar et al. [10]. The network flow is unrestricted in the sense that there are no prescribed nodes through which the flows should be re-routed, and there is a threshold-on-links based discounting for heavy traffic. Discounting incentives for amalgamation of flow lead to better utilization of high capacity links. This approach is certainly applicable to telecommunication networks with today’s explosion of bandwidth and speed.

† Requests for reprints should be sent to Jadranka Skorin-Kapov, W.A. Harriman School for Management and Policy, State University of New York at Stony Brook, Stony Brook, NY 11794-3775, USA.
It seems that threshold based discounting addresses some drawbacks of hub-networks. Namely, hub-networks involve a design where every two nodes have to communicate through a subset of nodes referred to as hubs (e.g. Campbell [3], Klincewicz [6], Ernst and Krishnamoorthy [5]). Sub-network consisting only of hub nodes is completely interconnected.

However, an analysis of hub-to-hub links utilization might reveal their disproportional flows, yet all hub-to-hub traffic is discounted. This is the motivation behind network design leading to discounting of only the appropriate, heavily used links. In O’Kelly and Bryan [8] the hub location problem was modified to include possibilities for differential discounts on interhub links, depending on total traffic amounts. A non-linear convex cost function was approximated by a piecewise linear function, which is in turn incorporated into a hub location problem.

In Podnar et al. [10] the hub concept was altogether abandoned and a new formulation was proposed with main emphasis on links of the network. In order to reach required thresholds for allowable discounting, the network users have to cooperate and amalgamate their flows. Sufficient amalgamation (> \( T \)) is rewarded, yielding reduction in the total flow cost. Possible application areas include:

- A fractional jet ownership with 2 types of aircraft, where threshold occurs if a single aircraft with larger capacity, can be used instead of a number of smaller ones.
- A small telecommunication company renting phone lines from a big one, where a discounting incentive is given for increased phone line utilization.
- A traveling agent buying air-plane seats for its customers, where the most popular destinations, with a significant demand, enjoy cheaper air fares.
- A shopper with a manufacturer’s coupon, where purchasing more items could generate additional savings per item purchased.

In Podnar et al. [10] the CAB (Civil Aeronautical Board) benchmark data set was used to test a computational approach based on branch and bounding. The approach delivered optimal solutions for smaller data instances (for 10 and 15 nodes). However, for data instances with 20 and 25 nodes, the computational requirements were prohibitively large and the obtained results were only suboptimal. The computational complexity of larger instances introduced the need to develop good heuristic solution approaches. In this paper we develop a genetic algorithm (GA) as a heuristic for cost minimization of networks with threshold based discounting.

Genetic Algorithms are based on observations of how living organisms pass the information to their offspring. In each cell of an organism there
is the same set of chromosomes. A chromosome consists of genes, encoded by a particular protein. A gene serves as a code for a trait (e.g., eye color). All possible choices for a trait (e.g., blue, green) are known as alleles. The set of all chromosomes is called the genome.

During reproduction, the genetic material from parents is combined in a crossover process. Newly generated chromosomes provide information for the offspring. Because of environmental factors, or because of the imperfection of the crossover process, a mutation of a gene can occur. The quality of the offspring, also known as the fitness, can be measured. An individual is fit if it has the ability to survive. This survival-of-the-fittest process can be mimicked in mathematical terms, and used in cases where search for the fittest means search for an optimal value.

The (GA) approach has been used to solve (optimally or approximately) a number of problems involving a combinatorial size explosion (NP-hard problems). The problems e.g. include: Traveling Salesman Problem (see Whitley, Starkweather and Shaner [12], Michalewicz [7]), Hub Location Problem (Abdennour-Helm [1], Abdennour-Helm and Venkataramanan [2]), Degree Constrained and Multi-Criteria Spanning Tree (Zhou and Gen [13], [14]), Maintenance Scheduling (Deris et al. [4]), and Uniform Graph Partitioning Problem (Pirkul and Rolland [9]).

In Section 2 we describe the problem and state mixed integer formulations for it. Section 3 presents a genetic search strategy adapted to our formulations. Computational results are displayed in Section 4. The paper concludes with some directions for further research.

2. Description and Formulations of the Problem

The problem that we address was formulated in Podnar et al. [10]. Given a completely interconnected network of physical links, the following assumptions are stated: (1) every pair of nodes is connected by a physical link represented by two directed links, (2) every directed link has been assigned a cost of sending a unit of traffic through the link, (3) every pair of network nodes must establish communication according to a given traffic flow matrix, (4) this communication generally will not follow a single path, i.e. the flow from source to sink can be split and sent via different routes; (5) the traffic can flow through any of the links, i.e. there is no restriction on communication only via a set of designated (say, hub-to-hub) links; (6) there is no upper limit on the number of intermediate nodes used to deliver traffic from a source to a destination; (7) there are no constraining link capacities; and (8) every node is capable of traffic rerouting and the increase
in time due to indirect traffic from a source to a sink (as opposed to direct flow) in negligible.

Following these assumptions, the problem is to decide which links can have discounted costs, based on the amount of traffic. The constant $T$ (the so-called threshold) is introduced so that if the flow through an edge is larger than the threshold $T$, then the cost of this flow is reduced by a factor $\alpha$.

As in Podnar et al. [10], let $N$ denote the set of nodes. For every pair of nodes $(k, m)$, $c_{km}$ is the cost of unit of flow going through the link $(k, m)$, and $(f_{ij})$ is the required amount of flow associated with every origin-destination pair $(i, j)$.

Given the assumptions, the objective is to find a feasible flow with minimal cost. We consider the total flow through a link, where any flow with the amount larger than the threshold $T$ is discounted. The complete set of variables used in the model is given as:

$$y_{km} = \begin{cases} 
0 & \text{link } (k, m) \text{ not discounted} \\
1 & \text{link } (k, m) \text{ discounted}
\end{cases}$$

$$x_{1}^{ij}_{km} = \text{not discounted flow from } i \text{ to } j \text{ through link } (k, m)$$

$$x_{2}^{ij}_{km} = \text{discounted flow from } i \text{ to } j \text{ through link } (k, m)$$

Before we continue with the formal definition of the problem, we proceed with an example to further clarify some aspects of the problem.

**Example 1:** In this example we show the solution dependency on the discount factor $\alpha$. Consider the network of three nodes as given in Figure 1a (c=costs; $f$=flow demands). The obtained solutions for the threshold $T = 3$ and three different values of $\alpha$ are given in Figure 1b. The discounted edges are represented with multiple lines, and the flow originating from node 1 to destination 2 is shown. We are required to send 2 units of flow from 1 to 2. The costs shown in the solutions are the discounted costs ($\alpha \cdot c$) where applicable.

We now continue with the problem formulation as given in Podnar et al. [10]:

$$\min \sum_{i,j,k,m:i\neq j, k \neq m, m \neq i, k \neq j} c_{km} \left(x_{1}^{ij}_{km} + \alpha x_{2}^{ij}_{km}\right) \quad (1)$$

s.t.

$$y_{km} T \leq \sum_{i,j,k,m:i \neq j, k \neq m, k \neq j} x_{2}^{ij}_{km} \quad k \neq m \quad (2)$$
GENETIC ALGORITHM FOR THRESHOLD BASED DISCOUNTING

Figure 1. a) An example of a network with specified costs $c$ and flow demands $f$; b) Sending 2 units of flow from origin 1 to destination 2, depending on three different values of $\alpha$. Threshold is $T = 3$.

$x_{ij}^{2j} \leq c_{km} \cdot f_{ij}$ \hspace{1cm} \{ i \neq j \hspace{0.5cm} k \neq m \hspace{0.5cm} m \neq i \hspace{0.5cm} k \neq j \}

(3)

$$\sum_{m,m\neq i} (x_{im}^{1j} + x_{2m}^{2j}) = f_{ij} \hspace{1cm} i \neq j$$

(4)

$$\sum_{k:k\neq l,k\neq j} (x_{kl}^{1j} + x_{2l}^{2j}) = \sum_{m,m\neq l,m\neq i} (x_{im}^{1j} + x_{2m}^{2j}) \hspace{0.5cm} l \neq i, j \hspace{0.5cm} i \neq j$$

(5)

$x_1, x_2 \geq 0 \hspace{1cm} y$ binary

The formulation has $2n(n-1)((n-2)^2 + (n-1)) = O(n^4)$ continuous, $n(n-1)$ binary variables, and $2n(n-1) + n(n-1)((n-2)^2 + (n-1)) + n(n-1)(n-2) = O(n^4)$ constraints.

Note that in a symmetric case ($c_{km} = c_{mk}$ $f_{ij} = f_{ji}$) it is possible to reduce the size of the program in half. The reduction is based on the following identities: $y_{km} = y_{mk}$ and $x_{2km}^{2j} = x_{2mk}^{1j}$ (the same for $x_1$). However, the size of the model is still $O(n^4)$. As the computational results
showed in Podnar et al. [10], the LP relaxation of this model is very tight, enabling good quality solutions by rounding.

However, for larger problems the size becomes restrictive. In Podnar et al. [10] a number of alternative models of size $O(n^3)$ has also been developed. The reduction from size $O(n^4)$ to size $O(n^3)$ is based on the fact that we are just interested in the amount of flow through the link $(k, m)$, hence we can disregard the destination of the incoming flow. Based on Podnar et al. [10] results, in this paper we employ the following transformation of our model to a model of size $O(n^3)$.

The size reduction will use this variable notation:

- $x_{1_{km}}^i$ = not discounted flow originating from $i$ through link $(k, m)$
- $x_{2_{km}}^i$ = discounted flow originating from $i$ through link $(k, m)$

Based on the above definition, the following identities hold: $x_{1_{km}}^i = \sum_j x_{1_{jm}}^{ij}$ and $x_{2_{km}}^i = \sum_j x_{2_{jm}}^{ij}$. Applying appropriate summations, the final model description can be formalized:

$$\min \sum_{i,k,m:i\neq m,k\neq m} c_{km} \left( x_{1_{km}}^i + \alpha x_{2_{km}}^i \right) \tag{6}$$

$$y_{km} T \leq \sum_{i:i\neq m} x_{2_{km}}^i \quad k \neq m \tag{7}$$

$$x_{2_{km}}^i \leq y_{km} \cdot \sum_{j:j\neq i} f_{ij} \quad k \neq m, i \neq m \tag{8}$$

$$\sum_{k,k\neq l} (x_{1_{kl}}^i + x_{2_{kl}}^i) - \sum_{m,i\neq m,m\neq l} (x_{1_{lm}}^i + x_{2_{lm}}^i) = f_{il} \quad l \neq i \tag{9}$$

$$x_1, x_2 \geq 0 \quad y \text{ binary}$$

This type of size reduction ($O(n^4) \rightarrow O(n^3)$) has been used in a number of hub-location problems. (See, e.g. Ernst and Krishnamoorthy [5].)

This formulation enjoys decrease in the number of variables, but suffers from a larger feasible set. Solutions to the $O(n^3)$ problem might violate the ‘capacity’ constraint (3), but the threshold constraint (2) will still be satisfied. In cases where the size is restrictive and it is not possible to obtain a solution to the $O(n^4)$ formulation, we will consider the solutions to the $O(n^3)$ formulation acceptable.

The formulation of size $O(n^3)$ is, in general, not tight. Hence, rounding will not provide good quality solutions. We need to resort to a heuristic improvement strategy. In the next section we present an adaptation of the GA approach as applicable to our problem.
3. GA Adaptation to Network Cost Minimization With Threshold Based Discounting

In this section we present basic elements of a binary genetic algorithm as adapted to our problem.

A binary genetic algorithm explores the set of feasible solutions, generating new ones from the old ones, by trying to improve their fitness. Solutions themselves are encoded into a binary sequence. Every binary sequence (i.e. possible solution) has a fitness assigned to it, representing the objective value for that particular solution.

In the (GA) terminology, the binary form of a possible solution will be referred to as a chromosome. Chromosomes are composed of genes, which might take on some number of values called alleles (in binary case = 0 or 1). The position of a particular gene is known as its locus.

The set of chromosomes that (GA) is exploring will be known as a generation. Usually the initial generation (sometimes called the initial population) consists of randomly chosen points in the solution space. In other words, the probability that a certain gene in a certain chromosome has value '1' is 50%. The Genetic Algorithm will go through the initial generation and modify it, yielding a next generation. We will denote generations with the symbol \( G(t) \), where \( t \) represents the cardinal number of the generation. \( G(0) \) will be the initial generation. In order to create next generations, (GA) will perform two types of operations: crossover operation (sometimes called recombination) and a mutation. A crossover usually selects two parent chromosomes from the current generation, and creates two offspring chromosomes. After reaching a desired level of fitness, the (GA) stops.

The adaptation of a (GA) to a specific problem involves making decisions regarding the following: what is the “best” pairing strategy; how to perform crossover; how frequently mutation should be initiated; how to select the initial population in order to improve the convergence.

Since the nature of our problem has a binary flavor, the representation of a feasible solution will be encoded using the binary variables \( y \). The fitness (objective value) can then be calculated by solving the linear program with \( y \) variables fixed. A preference will be given to the symmetric case \( c_{km} = c_{mk}, f_{ij} = f_{ji} \), but the same (GA) application can be easily modified to the non-symmetric case. The chromosomes will be written in this form:

\[
\text{chromosome } y_1, y_2, y_3, \ldots, y_{n-1}, y_n
\]
The number of bits in a chromosome ($N_{bits}$) in the symmetric case is $n(n - 1)/2$, where $n$ represents the number of nodes involved.

From the testing performed (see Podnar et al. [10]), it can be concluded that LP relaxation of size $O(n^4)$ is tight. Using binary values already obtained in the LP relaxation can provide a good start for the genetic algorithm. It has been noticed that in all solutions obtained by the LP relaxation of size $O(n^4)$, zeros in solutions to the LP relaxation stayed zeros in the optimal (MIP) solution. The binary variable which was one in the LP relaxation changed its value to zero in the optimal (MIP) solution very rarely (for $n = 10$ it occurred just twice considering all the threshold values $T$). Hence, one possibility for initial population selection would be to presolve the LP relaxation, and use the solution values which are already integer (0 or 1). It has been also noticed that fractional values close to 1 tend to achieve value 1 in the optimal (MIP) solution. (Similarly values close to 0 tend to achieve value 0.) Hence, after performing the LP relaxation, the solution values obtained will be used to generate the initial population. The size of the initial population will vary (depending on the size of the problem) and it will be denoted by $N_{init}$. The initial population is generated in the following way ($P$ stands for probability):

$$P(y_{km} = 1\text{ in initial population}) = P(y_{km} \text{ in LP relaxation})$$

For sizes of the problem exceeding the computability of the $O(n^4)$ version, the values in the initial population can be determined randomly. In other words, the probability of getting 0 or 1 is the same.

$$P(y_{km} = 1\text{ in initial population}) = 0.50$$

The rounding heuristic applied to the formulation of size $O(n^3)$ gave satisfactory results in the case of 25 nodes.

The rounding was based on LP relaxation of size $O(n^3)$ obtained by CPLEX solver with hybrid barrier option (which generated the alternative LP relaxation solution with more integer values). The LP relaxation solution can be used in the initial population:

$$P(y_{km} = 1\text{ in initial population}) = \begin{cases} y_{km} & \text{if } y_{km} = 0 \text{ or } 1 \text{ in the LP relaxation} \\ 0.50 & \text{otherwise} \end{cases}$$

We will generate relatively large initial population to provide the (GA) with a large sampling of the solution space. After sorting the chromosomes by
their fitness, the worst chromosomes are discarded, mimicking the natural selection process. To reduce the number of fitness evaluations that include solving large linear programs, the next populations will have smaller number of chromosomes. The number of chromosomes in a population (all but the initial one) will be denoted by $N_{\text{pop}}$.

The chromosomes are sorted by their objective function value, with the top-most chromosome being the fittest (in our case, the solution with the minimal objective value).

The fittest chromosomes in a population will be called “good” and, as “good” chromosomes, they will be used in the pairing procedure. The chromosomes with poor fitness will be called “bad” and they are to be replaced in the next population with new offspring.

On one hand, it is better to have more offspring (i.e. to explore larger sample subspace), but on the other, this implies more objective function evaluations, which might sometimes be very time and resource consuming. The “best” size must be determined experimentally.

The rate between $N_{\text{good}}$ and $N_{\text{pop}}$ is called the crossover rate ($X_{\text{rate}}$):

$$N_{\text{good}} = X_{\text{rate}} N_{\text{pop}}.$$

The pairing (selection of chromosomes for crossover), will depend on the fitness of a chromosome. The best chromosomes are chosen more frequently. For each “good” chromosome, a normalized cost is calculated, subtracting the cost of the first “not-good” chromosome from the cost of all the chromosomes in the ’mating pool’ (i.e. “good” ones). If $l$ is the $l^{th}$ best chromosome, then the normalized cost can be written:

$$C_l = \text{cost}_l - \text{cost}_{N_{\text{good}}+1}.$$

Then the probability of choosing the $l^{th}$ chromosome for a crossover is:

$$P(l = \text{a parent}) = \frac{C_l}{\sum_{p=1}^{N_{\text{good}}} C_p}.$$

If there is large spread in costs between the $1^{st}$ and $N_{\text{good}}+1^{th}$ chromosome, the best chromosome will be more favorable. If the chromosomes have approximately the same costs, then the probabilities will also be the same.

When two different chromosomes are selected, a crossover operation can be performed.

The crossover operation starts with the two selected chromosomes (parents) and generates two new ones, which will then replace two “bad” chromosomes. The crossover operation is a uniform crossover, in which a random binary mask is formed (zeroes and ones are assigned randomly). Based on the binary mask, the first offspring will inherit either 1$^{st}$ parent’s gene (mask=0) or 2$^{nd}$ parent’s gene (mask=1). The genes that are shared by
both parents will be shared by the offspring too. So, in the case where the LP relaxation is used, the original structure of zeros is passed on to the next generation.

Now, a mutation can occur. Random mutations alter a small percentage of bits in the current generation. The mutation is expected to generate solutions farther from the original generation, in order to explore new regions and to avoid local optima. This will prevent the (GA) from converging (and stopping) too fast. Mutation points are randomly selected from the current population (which has $N_{\text{bits}} \times N_{\text{pop}}$ bits). The rate of mutation will be denoted by $\mu$. Hence number of changed bits is $\mu \cdot N_{\text{bits}} \times N_{\text{pop}}$.

Mutation will not be performed on the best solution. (The solutions that are left out from the mutation range are designated as elite solutions.) This will ensure the propagation of the best solutions in the next population.

In the cases where the LP relaxation is used, the mutation will not be induced on the bits that are zeros in the LP relaxation. Sometimes (but very rarely) there is a need to change a bit which corresponds to 1 in the LP relaxation, into 0. The remaining bits are also open for mutations. Most mutations raise the cost of a chromosome (i.e. its fitness is lowered). The occasional lowering of the costs adds diversity into the population.

After a new generation has been created, the (usually very expensive) evaluations of the objective function can take place. The new population of chromosomes will be sorted in increasing order according to their fitness.

The stopping criteria will be based on the objective value of the best chromosome, on the behavior of the average value, and on standard deviation for the objective values in the current generation. If all the chromosomes have the same objective value as the $N_{\text{good}} + 1$ chromosome, then the (GA) cannot calculate the pairing probabilities, and hence will be forced to stop (the standard deviation is 0). The maximum number of generation will be preset and denoted by MaxGen.

The pseudo-code of the algorithm is given in Figure 2.

4. Computational Results

Tests were performed on the same data set as in Podnar et al. [10]. Performance of the (GA) can, then, be compared with the results generated by their branch and bound technique.

The data set used in Podnar et al. [10] is known as the Civil Aeronautical Board (CAB) data set. The CAB data set consists of air passenger traffic in the United States in 1970. 25 cities are represented with the heaviest air passenger traffic. Subproblems of 10, 15, 20 and 25 nodes have been used in a number of hub related studies.
Figure 2. Pseudo-code

**INITIALIZATION**

○ Step 0. Read the data and initialize the (GA) parameters: \( N_{\text{init}}, N_{\text{pop}}, N_{\text{good}}, N_{\text{bad}}, \mu, \text{MaxGen}, \text{NumRuns} \)

○ Step 1. Evaluate the initial LP relaxation using either \( O(n^3) \) or \( O(n^4) \) formulation.

○ Step 2. Based on the LP relaxation’s optimal solution calculate CMutation (cumulative mutation probabilities); calculate the number of mutation bits (NumMut).

○ Step 3. Generate the initial population (\( N_{\text{init}} \)) using the LP relaxation solution according to one of the following strategies: LP4 (\( O(n^4) \)) or LP3 (\( O(n^3) \)). (Explanation of LP4 and LP3 notation is given in the section 4.)

○ Step 4. Sort the initial population in increasing order using members’ fitness values. Calculate the cumulative pairing probabilities. Print the best individual and the statistics (average fitness, standard deviation).

**REPEAT NumRuns TIMES:**

• Step 5. Set the seed value for the random number generator.

**REPEAT UNTIL STOPPING CRITERION IS SATISFIED**

(Stopping criterion checks if the number of generations reached the maximum number of generations, and if all the members of the current population differ or if they are all the same.)

• Step 6. Perform the pairing procedure based on the current population and the cumulative pairing probabilities.

• Step 7. Perform the mutation (based on the cumulative mutation probabilities) NumMut times.

• Step 8. Evaluate the new members’ costs.

• Step 9. Sort the current population.

• Step 10. Calculate new cumulative pairing probabilities.

• Step 11. Print the best individual and the statistics (average fitness, standard deviation).

• Step 12. If the stopping criteria is not satisfied return to Step 6.

• Step 13. If the current number of runs is less than NumRuns return to Step 5 with the different starting seed value.

○ Step 14. Display the best solution and the time when it was found.
Problems with small sizes \((n=10,15)\) can be solved to optimality in an acceptable time frame by the branch and bound approach. Our main concern was to find heuristic solutions to the problems with larger number of nodes \((n=20,25)\).

The Genetic Algorithm code was written in C. In the cases where the LP relaxation was needed, the model of size \(O(n^4)\) was used when possible (for \(n = 10, 15, 20\)). For \(n = 25\) we employed the formulation of size \(O(n^3)\). In all instances, the individual fitness was evaluated by means of the smaller \((O(n^3))\) model. Solutions to all linear programs were obtained by calls to the CPLEX®6.0 callable library. Different CPLEX® options were used in order to minimize the time consumption. For the \(O(n^4)\) cases, the dual method was used; for the LP relaxations of size \(O(n^3)\), the hybrid barrier method resulted in the solutions with more integer entries. (For some instances the primal method worked faster.) The initial LP relaxation was used to calculate the probabilities \(P(y_{km} = 1)\).

The LP relaxation was also used to plant seeds in the initial population. The seeds are rounded solutions obtained from the LP relaxations. Integer values of decision variables were set to 1 if the corresponding \(y_{km}\) in the LP relaxation solution was greater than some predetermined numerical scale. For size \(O(n^4)\), the scales used were: 0.50, 0.65, 0.75 and 0.85. For size \(O(n^3)\), the 0.50 and 0.65 scales were used. The rest of the initial population was determined in the usual way (based on the obtained probabilities).

All tests were run on SUN SPARC-stations. Random number generator \((\text{drand48()})\) was initialized with the same seed through all the runs. The best and the average solutions were monitored, together with the standard deviation of the objective values in the current generation.

Based on the initial LP relaxation size, the runs were partitioned into two “strategic” categories. The strategies and their short explanations are given in the following chart:

<table>
<thead>
<tr>
<th>LP4</th>
<th>LP relaxation model: size (O(n^4)) (scales: 0.50, 0.65, 0.75, 0.85)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP3</td>
<td>LP relaxation model: size (O(n^3)) (scales: 0.50, 0.65)</td>
</tr>
</tbody>
</table>

The number of all generations evaluated in the particular run, together with the first generation where the best (and final) solution has occurred, was recorded.

Initial tests were performed without checking whether the two selected chromosomes for crossover are the same or not. Preliminary testing was performed with different population sizes and different mutation rates, in order to get information about the most suitable parameter values, leading to improved performance. “Large” mutation rate increased variety in the populations, but it disturbed the convergence. On the other hand, “small”
The population size should not be too large, because of the expensive fitness evaluations. However, the size should not be too small either, limi-
Preliminary Results: Genetic Algorithm Strategy; size $O(n^4)$

<table>
<thead>
<tr>
<th>$\mu = 0.10$</th>
<th>$N_{\text{init}} = 32$</th>
<th>$N_{\text{pop}} = 16$</th>
<th>$N_{\text{good}} = 8$</th>
<th>$N_{\text{had}} = 8$</th>
<th>MaxGen=20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 10$</td>
<td>$n = 15$</td>
<td>$n = 20$</td>
<td>$n = 10$</td>
<td>$n = 15$</td>
<td>$n = 20$</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>$1^\text{st}$</td>
<td>gap IP</td>
<td>all</td>
<td>$1^\text{st}$</td>
</tr>
<tr>
<td>frv</td>
<td>mvs</td>
<td>gap b&amp;b IP</td>
<td>time</td>
<td>time</td>
<td>obj</td>
</tr>
<tr>
<td>5K</td>
<td>10K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>20K</td>
<td>20K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>15K</td>
<td>15K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>30K</td>
<td>30K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>40K</td>
<td>40K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>50K</td>
<td>50K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>60K</td>
<td>60K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>70K</td>
<td>70K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>80K</td>
<td>80K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
<tr>
<td>90K</td>
<td>90K</td>
<td>0</td>
<td>0</td>
<td>6frv</td>
<td>1</td>
</tr>
</tbody>
</table>

### Notes:
- frv: branch and bound
- mvs: all number of generations in the run
- $1^\text{st}$ generation at which the best (GA) solution has occurred
- gap IP: $\text{obj}_{\text{LP}} - \text{obj}_{\text{IP}}$ = optimal objective value to the integer program (Podnar et al. [10])
- gap b&b IP: $\text{obj}_{\text{b&b IP}} - \text{obj}_{\text{IP}}$ = best known objective value to the integer program obtained by b&b (Podnar et al. [10])
- gap LP: $\text{obj}_{\text{LP}} - \text{obj}_{\text{IP}}$ = optimal objective value to the LP relaxation (Podnar et al. [10])
- n: number of generations in the run (rule=1) (Podnar et al. [10])
- $1^\text{st}$ time from the b&b run (rule=1) (Podnar et al. [10])
- IP = optimal objective
- LP = optimal objective value to the LP relaxation (Podnar et al. [10])
- *: IP known to be optimal

The preference was given to $N_{\text{init}} = 30$, $N_{\text{pop}} = 25$, $N_{\text{good}} = 10$. Maximum number of generations was chosen so that the runs could be compared with the branch and bound solutions.
The optimality was obtainable in the cases of small $n$ ($n = 10, 15$). In these cases, branch and bound approach was preferred to the (GA) approach since its solutions were proved to be optimal. The use of (GA) was expected to improve the solutions for larger number of nodes (either by value, or by time, or both).

Table 1 presents optimal solution values for $n = 10, 15$, and the best known branch and bound solutions for $n = 20, 25$, as obtained from Podnar et al. [10]. Times to get the solutions are also displayed.

Table 2 presents preliminary results for $n = 10$, $n = 15$ and $n = 20$ nodes. For $n = 10$, the (GA) terminated in majority of cases with the initial population, due to a population with almost all individuals being the same. This is not a surprise, because it has been shown in Podnar et al. [10] that the LP relaxation of size $O(n^4)$ is very tight. Also, for $n = 10$ the rounding of the LP relaxation gave the optimal solution in almost all cases with different thresholds. The times are very similar, but the (GA) does not guarantee that the obtained solution is the optimal one. For $n = 15$ case, all the runs needed all 20 available populations. Again, it is visible that the best solution is obtained early in the run. The (GA) performed just the random search ((GA)’s initial population giving the best solution), but even in this case it generated satisfactory results (the gap between the (GA) objective value and the best lower bound obtained by a branch and bound strategy was always $<0.20\%$), in ten or twenty minutes of CPU time. For $n = 20$ case, the optimal solutions were not available. Again the (GA) performed just the random search and the best solution was obtained in the initial population. Even in this case, the final solution was within $<0.331\%$ from the LP relaxation, and within $<0.208\%$ from the best known solution. The improvement over the branch and bound best solutions was made in 8 out of 18 cases, and within a couple of hours of running time. Branch and bound solutions needed more time even without the optimality proof (see Table 1). The random search could be performed within 1 hour and 15 min, with satisfactory results, often outperforming the branch and bound results regarding objective function values.

The random search did not employ all the benefits of the Genetic Algorithm approach. In other words, the crossover and the mutation were ineffective. To avoid crossovers between the same individuals (those crossovers are actually just reproductions), the testing was performed before the actual crossover took place. This testing was just checking if the objective values (fitness) of the two selected individuals were the same or not. Checking all the bits was not used, since it would increase the complexity of the crossover $N_{\text{bits}}$ times. A procedure was used to stop the pure reproductions.
The improved results are listed in Tables 3, 4 and 6. For \( n = 15 \) (Table 3) the (GA) produced results within a reasonable time frame (<23min for the worst case). The final solutions were equal to the optimal solutions (obtained by branch and bound) in 7 out of 18 runs. In other cases the gap did not go over 0.11335% in the most difficult case. The results of the branch and bound strategy \texttt{frv:1} from Podnar et al. [10] were gathered for the comparison. The \texttt{frv:1} strategy selects the vertex with the minimal number of fractional variables and, if tie, minimal objective value. The (GA) evolved near optimal solutions very quickly. The solutions were recognized very early in the evolution of the populations. Mutation rate of 0.02 worked better than 0.01, 0.03, 0.05 and 0.10 rates.

Similar results were observed in the case \( n = 20 \). The list of the best branch and bound solutions was used for comparison purposes. In a couple of hours, the (GA) produced very good results. The branch and bound technique proved optimality of our problem for \( T = 5K \) and \( T = 90K \). The optimal solutions were also found by the (GA) approach. The (GA) improved the branch and bound results in majority of cases (11 out of 18, plus two optimal, plus two cases with the same solutions). The (GA) runs were very efficient in their time consumption.

Regarding the results for \( n = 25 \), the (GA) improved the branch and bound based heuristic. This was expected, since the heuristic is a rounding one. However, the time consumption turned out to be very demanding. Hence, the maximum number of generations was decreased from 15 to 10. The results with MaxGen=10 are presented in Table 4. It was noticed that the standard deviation of the objective values in a single generation increased rapidly, giving a hint of diversifying the population. Also the average value started to increase. The adjustment of parameters was necessary. Based on those observations, the mutation rate was decreased from 0.02 to 0.01. The results (\( \mu = 0.01 \)) are shown in the Table 4. Decrease of the mutation rate enabled the algorithm to converge faster. The gaps were decreased, together with the times. It was encouraging to decrease the mutation rate further. The results with the rate 0.005 are also shown in Table 4. Finally, Table 6 lists results with increased number of generations (MaxGen=60). Further improvements in solution quality were observed, but the time consumption was considerably increased. Balancing the convergence and exploration of new regions by means of parameter changes increased the number of populations necessary for obtaining the final solutions. If the best solution did not improve even after 10 successive generations the algorithm would stop.
Table 3. Genetic Algorithm Strategy; size $O(n^4)$

<table>
<thead>
<tr>
<th>$n$</th>
<th>all $\text{time}$ (s)</th>
<th>$\text{fri} : 1$</th>
<th>gap $\text{IP}^*$</th>
<th>all $\text{time}$</th>
<th>$\text{fri} : 1$</th>
<th>gap $\text{b&amp;b IP}^*$</th>
<th>all $\text{time}$</th>
<th>$\text{fri} : 1$</th>
<th>gap $\text{LP}^*$</th>
<th>all $\text{time}$</th>
<th>$\text{fri} : 1$</th>
<th>gap $\text{LP}^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>5K 6m18s 1m28s 2m54s 24m59s</td>
<td>15</td>
<td>0</td>
<td>0.00053%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>10K</td>
<td>8m51s 2m05s 17m31s 32m39s</td>
<td>15</td>
<td>0</td>
<td>0.01197%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>15K</td>
<td>13m23s 2m24s 1h04m23s 47m03s</td>
<td>15</td>
<td>0</td>
<td>0.01135%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>20K</td>
<td>14m40s 3m47s 1h24m06s 54m45s</td>
<td>15</td>
<td>0</td>
<td>0.00087%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>25K</td>
<td>15m26s 1m31s 2h10m17s 5m61s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>30K</td>
<td>17m57s 9m31s 50m27s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>35K</td>
<td>18m54s 15m51s 3h44m13s 40m57s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>40K</td>
<td>22m24s 12m50s 1h59m46s 4h34m35s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>45K</td>
<td>25m14s 21m35s 1h39m15s 1h17m29s 47m36s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>50K</td>
<td>27m21s 6m24s 42m02s 4h8m37s 1h07m07s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>55K</td>
<td>29m11s 8m13s 19m26s 4h48m35s 1h04m07s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>60K</td>
<td>31m18s 8m48s 7m55s 4h28m25s 1h21m17s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>65K</td>
<td>33m06s 5m59s 7m29s 4h51m58s 1h43m40s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>70K</td>
<td>36m36s 7m00s 14m02s 1h23m40s 1h52m01s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>75K</td>
<td>38m57s 7m24s 7m21s 2h04m36s 1h34m33s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>80K</td>
<td>41m14s 8m08s 14m18s 2h12m40s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>85K</td>
<td>43m34s 5m18s 16m11s 4h48m36s 1h33m51s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>90K</td>
<td>46m05s 7m43s 25m48s 4h41m56s 1h28m22s</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
<td>0.00000%</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

b&b: branch and bound
all: number of generations in the run
$f^*: 1^{st}$ generation at which the best (GA) solution has occurred
$\text{gap IP}^*: \text{obj}_{\text{IP}} - \text{best} (\text{GA}) \text{ solution}; \text{IP} = \text{optimal objective value} \text{ to the integer program (Podnar et al. [10])}
\text{gap b&b IP}^*: \text{obj}_{\text{b&b IP}} - \text{best known objective value} \text{ to the integer program obtained by b&b (Podnar et al. [10])}
\text{gap LP}^*: \text{obj}_{\text{LP}} - \text{obj}_{\text{IP}}; \text{LP} = \text{optimal objective value} \text{ to the LP relaxation (Podnar et al. [10])}
$n/a$: no gap (obj=IP)
$fri$: times from the b&b-fri run (rules=1) (Podnar et al. [10])
$\ast$: IP known to be optimal

For the benchmark data tested, the population size of 25 chromosomes was a reasonable choice. Bigger generations increased the evaluation times, while smaller ones caused the early termination of the algorithm.
Table 4. Genetic Algorithm Strategy: size $O(n^3)$

<table>
<thead>
<tr>
<th>n = 25</th>
<th>N_{out} = 30</th>
<th>N_{gap} = 25</th>
<th>N_{b&amp;b} = 10</th>
<th>N_{had} = 15</th>
<th>MaxGen = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>1st</td>
<td>all</td>
<td>1st</td>
<td>all</td>
</tr>
<tr>
<td>$\mu$</td>
<td>time (s)</td>
<td>gap b&amp;b IP</td>
<td>time (s)</td>
<td>gap b&amp;b IP</td>
<td>time (s)</td>
</tr>
<tr>
<td>10</td>
<td>2h39m26s</td>
<td>52n56b5s</td>
<td>1h23m48s</td>
<td>4h15m04s</td>
<td>2h17m15s</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-0.0071%</td>
<td>10</td>
<td>0.0009%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>-0.0044%</td>
<td>-10</td>
<td>-0.0086%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>-0.0062%</td>
<td>10</td>
<td>-0.1488%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>-0.0172%</td>
<td>10</td>
<td>-0.1031%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>-0.3316%</td>
<td>10</td>
<td>-0.1918%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>-0.2016%</td>
<td>7</td>
<td>-0.3444%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>-0.3496%</td>
<td>10</td>
<td>-0.3496%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>-0.3316%</td>
<td>10</td>
<td>-0.2016%</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>-0.3496%</td>
<td>10</td>
<td>-0.3496%</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>-0.3316%</td>
<td>0</td>
<td>-0.2016%</td>
<td>0</td>
</tr>
</tbody>
</table>

b&b: branch and bound  
all: number of generations in the run  
1st: 1st generation at which the best (GA) solution has occurred  
gap b&b: gap between the branch and bound (b&b) solution and the best known (GA) solution (Podnar et al. [10])  
gap LP: gap between the best solution to the linear program (LP) and the best known solution to the integer program (IP).
GENETIC ALGORITHM FOR THRESHOLD BASED DISCOUNTING

Table 5. Summary of results \( n = 25 \), \( \mu = 0.005 \), \( N_{\text{init}} = 30 \), \( N_{\text{pop}} = 25 \), \( N_{\text{good}} = 10 \), \( N_{\text{bad}} = 15 \)

<table>
<thead>
<tr>
<th>T</th>
<th>MaxGen=10</th>
<th>MaxGen=60</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>improvement over branch and bound</td>
</tr>
<tr>
<td>10K</td>
<td>2.75h</td>
<td>0.133%</td>
</tr>
<tr>
<td>20K</td>
<td>2.88h</td>
<td>0.113%</td>
</tr>
<tr>
<td>30K</td>
<td>3.19h</td>
<td>0.536%</td>
</tr>
<tr>
<td>40K</td>
<td>3.06h</td>
<td>0.342%</td>
</tr>
<tr>
<td>50K</td>
<td>3.26h</td>
<td>0.495%</td>
</tr>
<tr>
<td>60K</td>
<td>3.44h</td>
<td>0.445%</td>
</tr>
<tr>
<td>70K</td>
<td>3.24h</td>
<td>0.587%</td>
</tr>
<tr>
<td>80K</td>
<td>3.19h</td>
<td>0.483%</td>
</tr>
<tr>
<td>90K</td>
<td>3.06h</td>
<td>0.389%</td>
</tr>
</tbody>
</table>

5. Conclusions and Future Work

In this paper a Genetic Algorithm (GA) solution approach for cost minimization as applied to networks with threshold based discounting was presented, and compared to previous branch and bound results.

When considering problem instances with small values of \( n \) (\( n = 10, 15 \)), the branch and bound procedure generates optimal solutions quickly. In those cases the (GA) approach might give close-to-optimal results, but the branch and bound technique is preferable.

In the cases where \( n \) is larger (\( n = 20, 25 \)) the branch and bound procedure fails to obtain optimal solutions in a reasonable time due to a significant increase in the size of the problem. Branch and bound tree grows exponentially, which makes a solution searches very ineffective. On the other hand, (GA) experiences a linear increase in the size of its data structures (population). Based on our results, the use of (GA) in large-sized problems is preferred.

Selection of (GA) parameters was dependent on experimental results. An increase in the mutation rate would broaden the search space, hence increasing the odds of finding the global optimum. However, the effect of such an increase might be very slow convergence of the algorithm. If the size of population is not big enough, the search could end up in a local optimum in early stages of the algorithm. On the other hand, big population should be avoided, due to very expensive evaluations of the objective function.

The algorithm performance also depended on the size of the LP relaxation used. In the cases where LP relaxation of size \( O(n^4) \) was possible to evaluate, the search space was “smaller” due to the tightness of the relaxation. In these cases “bigger” mutation rate might be more appropriate in
Table 6. Results : Genetic Algorithm Strategy; size $O(n^3)$

<table>
<thead>
<tr>
<th>$n = 25$</th>
<th>$\mu = 0.005$</th>
<th>$N_{init} = 30$</th>
<th>$N_{pop} = 25$</th>
<th>$N_{good} = 10$</th>
<th>$N_{bad} = 15$</th>
<th>MaxGen=60</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time 1st</td>
<td>time obj</td>
<td>gap b&amp;b IP</td>
<td>gap LP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td>25</td>
<td>8h16m48s</td>
<td>$7,493,749,899.43$</td>
<td>$-0.00043%$</td>
<td>$0.03591%$</td>
<td></td>
</tr>
<tr>
<td>10K</td>
<td>10h02m36s</td>
<td>8h19m48s</td>
<td>$7,501,767,383.89$</td>
<td>$-0.20666%$</td>
<td>$0.14715%$</td>
<td></td>
</tr>
<tr>
<td>15K</td>
<td>10h24m29s</td>
<td>8h25m32s</td>
<td>$7,512,960,034.28$</td>
<td>$-0.33800%$</td>
<td>$0.29657%$</td>
<td></td>
</tr>
<tr>
<td>20K</td>
<td>13h05m34s</td>
<td>13h05m34s</td>
<td>$7,526,910,253.30$</td>
<td>$-0.36933%$</td>
<td>$0.48280%$</td>
<td></td>
</tr>
<tr>
<td>25K</td>
<td>14h02m48s</td>
<td>14h02m48s</td>
<td>$7,535,000,011.50$</td>
<td>$-0.71305%$</td>
<td>$0.59080%$</td>
<td></td>
</tr>
<tr>
<td>30K</td>
<td>13h05m10s</td>
<td>10h15m08s</td>
<td>$7,555,677,024.14$</td>
<td>$-0.82606%$</td>
<td>$0.86683%$</td>
<td></td>
</tr>
<tr>
<td>35K</td>
<td>15h10m30s</td>
<td>12h34m27s</td>
<td>$7,566,292,197.33$</td>
<td>$-1.08845%$</td>
<td>$1.00854%$</td>
<td></td>
</tr>
<tr>
<td>40K</td>
<td>11h59m52s</td>
<td>9h16m01s</td>
<td>$7,583,474,002.89$</td>
<td>$-0.86252%$</td>
<td>$1.23792%$</td>
<td></td>
</tr>
<tr>
<td>45K</td>
<td>19h26m42s</td>
<td>12h17m08s</td>
<td>$7,589,534,633.00$</td>
<td>$-0.9845%$</td>
<td>$1.31882%$</td>
<td></td>
</tr>
<tr>
<td>50K</td>
<td>15h00m48s</td>
<td>10h48m26s</td>
<td>$7,602,461,197.92$</td>
<td>$-1.03555%$</td>
<td>$1.49139%$</td>
<td></td>
</tr>
<tr>
<td>55K</td>
<td>19h06m42s</td>
<td>12h17m08s</td>
<td>$7,614,835,764.07$</td>
<td>$-0.96471%$</td>
<td>$1.65659%$</td>
<td></td>
</tr>
<tr>
<td>60K</td>
<td>10h48m26s</td>
<td>9h16m01s</td>
<td>$7,617,315,069.38$</td>
<td>$-0.96743%$</td>
<td>$1.69049%$</td>
<td></td>
</tr>
<tr>
<td>65K</td>
<td>11h04m17s</td>
<td>11h04m17s</td>
<td>$7,629,488,742.44$</td>
<td>$-0.7252%$</td>
<td>$1.98570%$</td>
<td></td>
</tr>
<tr>
<td>70K</td>
<td>14h02m48s</td>
<td>12h17m08s</td>
<td>$7,639,488,742.41$</td>
<td>$-0.84117%$</td>
<td>$2.26460%$</td>
<td></td>
</tr>
<tr>
<td>75K</td>
<td>15h04m59s</td>
<td>11h04m17s</td>
<td>$7,660,380,154.29$</td>
<td>$-0.66117%$</td>
<td>$2.54997%$</td>
<td></td>
</tr>
<tr>
<td>80K</td>
<td>9h12m30s</td>
<td>5h34m01s</td>
<td>$7,665,993,622.92$</td>
<td>$-0.92569%$</td>
<td>$2.33954%$</td>
<td></td>
</tr>
<tr>
<td>85K</td>
<td>17h33m42s</td>
<td>12h07m40s</td>
<td>$7,681,756,626.93$</td>
<td>$-0.66117%$</td>
<td>$2.54997%$</td>
<td></td>
</tr>
<tr>
<td>90K</td>
<td>7h11m36s</td>
<td>4h17m33s</td>
<td>$7,687,965,101.07$</td>
<td>$-0.66117%$</td>
<td>$2.54997%$</td>
<td></td>
</tr>
</tbody>
</table>

b&b: branch and bound

all: number of generations in the run

1st: 1st generation at which the best (GA) solution has occurred

gap LP: $\frac{obj - LP}{LP}$, obj = best (GA) solution; LP = optimal objective value to the LP relaxation (Podnar et al. [10])
gap b&b IP: $\frac{b&b IP - obj}{b&b IP}$, b&b IP = best known objective value to the integer program obtained by b&b (Podnar et al. [10])

order to jump into a new region. This might be very useful in the cases where a decision variable $y_{km}$ has value 1 in the LP relaxation, but changes its value to 0 in the optimal solution.

In the cases where LP relaxation of size $O(n^4)$ could not be exploited due to a high memory consumption, the LP relaxation of size $O(n^3)$ could be used. Based on the results with the branch and bound technique, the formulation of size $O(n^3)$ proved to be not very tight (especially when the threshold $T$ increases). As a consequence, the solution space is “larger” than in the $O(n^4)$ cases. Hence, smaller mutation rate might outperform the bigger ones. This observation is supported by the experimental results (Tables 4,6).

Based on the CAB data set and the presented results, our suggestion is use of (GA) strategy for larger values of $n$ ($n = 25$) with the following (GA) parameters: $\mu = 0.005$ $N_{init} = 30$ $N_{pop} = 25$ $N_{good} = 10$ $N_{bad} = 15$. A concise summary of the results ($n = 25$), indicating effectiveness of the
(GA) approach over branch and bound techniques, can be viewed in the tabular format (see Table 5).

For each of the thresholds ranging from $T=10K$ to $T=90K$ in 10K intervals, the time necessary for (GA) runs is recorded. If the comparison of the (GA) and the branch and bound results is concerned, improvement in the objective value is measured by the gap between the best (GA) solution and the best known objective obtained by the branch and bound techniques. The percentage is obtained relative to the best known branch and bound solution. (For detailed results one should refer to Tables 4,6.) The performance of the (GA) approach might be improved utilizing different genetic operators. Future work could introduce a time factor into the operators’ performance, such as non-uniform mutations (the rate decreases over time). Another possibility is to develop a strategy with variable population size.

Crossover operation might be altered, i.e. it is possible to use the original costs instead of normalized ones, or to define the probabilities based solely on rank of the chromosomes. Also, the fitness can be re-evaluated according to some adjustment function (e.g. $\frac{1}{1+x^2}$), and the new composed value could be used in the selection process.

The problem formulation might undergo several changes. Link capacities could be introduced, making the problem tighter. Also, the threshold $T$ could be modified to reflect specific needs on certain links. In other words, a matrix $T = T_{km}$ could be introduced.

Acknowledgments

The work was partially supported by the NSF grants SBR-9602021 and ANIR-9814014, and by the project 036033 - Architectural Elements for Regional Information Infrastructures, funded jointly by the Ministry of Science and Technology of the Republic of Croatia and the Istrian County.

References


