JOURNAL OF APPLIED MATHEMATICS AND DECISION SCIENCES, 7(4), 207–228

Genetic Algorithm for Network Cost Minimization Using Threshold Based Discounting

HRVOJE PODNAR podnar@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 α. 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]). Subnetwork 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 (Abdinnour-Helm [1], Abdinnour-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 α.

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:

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

$$\begin{aligned} x1^{ij}_{km} &= \text{ not discounted flow from } i \text{ to } j \text{ through link } (k,m) \\ x2^{ij}_{km} &= \text{ discounted flow from } i \text{ to } j \text{ through link } (k,m) \quad . \end{aligned}$$

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

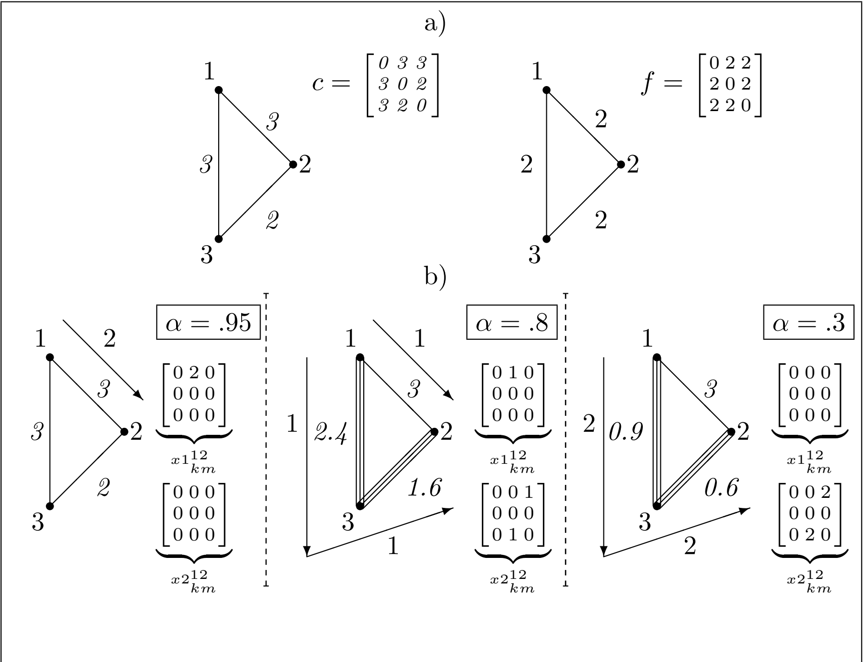
Example 1: In this example we show the solution dependency on the discount factor α. Consider the network of three nodes as given in Figure 1a (c=costs; f=flow demands). The obtained solutions for the threshold $T = 3$ and three different values of α are given in Figure 1b. The discounted edges are represented 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]:

$$\min \sum_{\substack{i,j,k,m: i\neq j, \\ k\neq m, m\neq i, k\neq j}} c_{km} \left(x1^{ij}_{km} + \alpha x2^{ij}_{km} \right) \tag{1}$$

s.t.

$$y_{km} T \leq \sum_{\substack{i,j: i\neq j, \\ i\neq m, j\neq k}} x2^{ij}_{km} \qquad k \neq m \tag{2}$$



a)
$c = \begin{bmatrix} 0 & 3 & 3 \\ 3 & 0 & 2 \\ 3 & 2 & 0 \end{bmatrix}$
$f = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$
b)
$\alpha = .95$
$\alpha = .8$
$\alpha = .3$
$x1^{12}_{km}$
$x2^{12}_{km}$
2.4
1.6
0.9
0.6

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 α. Threshold is $T = 3$.

$$x2^{ij}_{km} \le y_{km} \cdot f_{ij} \qquad \begin{cases} i \neq j & k \neq m \\ m \neq i & k \neq j \end{cases} \tag{3}$$

$$\sum_{m:m\neq i} \left(x1^{ij}_{im} + x2^{ij}_{im} \right) = f_{ij} \qquad i \neq j \tag{4}$$

$$\sum_{k:k\neq l, k\neq j} \left(x1^{ij}_{kl} + x2^{ij}_{kl} \right) = \sum_{m:m\neq l, m\neq i} \left(x1^{ij}_{lm} + x2^{ij}_{lm} \right) \quad l \neq i,j \ , \ i \neq j \tag{5}$$

$$x1, x2 \geq 0 \quad y \text{ binary}$$

The formulation has $2n(n-1)\left((n-2)^2 + (n-1)\right) = O(n^4)$ continuous, $n(n-1)$ binary variables, and $2n(n-1) + n(n-1)\left((n-2)^2 + (n-1)\right) + n(n-1)(n-2) = O(n^4)$ 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 $x2^{ij}_{km} = x2^{ji}_{mk}$ (the same for $x1$). 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:

$$\begin{aligned} x1^i_{km} &= \text{ not discounted flow originating from } i \text{ through link } (k,m) \\ x2^i_{km} &= \text{ discounted flow originating from } i \text{ through link } (k,m) \quad . \end{aligned}$$

Based on the above definition, the following identities hold: $x1^i_{km} = \sum_j x1^{ij}_{km}$ and $x2^i_{km} = \sum_j x2^{ij}_{km}$. Applying appropriate summations, the final model description can be formalized:

$$\min \sum_{i,k,m:i\neq m,k\neq m} c_{km}\left(x1^i_{km} + \alpha x2^i_{km}\right) \tag{6}$$

$$y_{km}T \leq \sum_{i:i\neq m} x2^i_{km} \qquad k \neq m \tag{7}$$

$$x2^i_{km} \leq y_{km} \cdot \sum_{j:j\neq i} f_{ij} \qquad k \neq m, i \neq m \tag{8}$$

$$\sum_{k:k\neq l}\left(x1^i_{kl} + x2^i_{kl}\right) - \sum_{m:i\neq m,m\neq l}\left(x1^i_{lm} + x2^i_{lm}\right) = f_{il} \quad l \neq i \tag{9}$$

$$x1, x2 \geq 0 \qquad y \text{ binary}$$

This type of size reduction $(O(n^4) \to O(n^3))$ 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:

$$\textit{chromosome} \qquad \underline{y_{12}}\ \underline{y_{13}}\ \underline{y_{14}}\ \ldots\ \underline{y_{n-1n}}$$

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):

$$P(\underbrace{y_{km} = 1}_{\substack{\text{in initial}\\ \text{population}}}) = \underbrace{y_{km}}_{\substack{\text{in LP}\\ \text{relaxation}}} \quad .$$

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.

$$P(\underbrace{y_{km} = 1}_{\substack{\text{in initial}\\ \text{population}}}) = 0.50 \quad .$$

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:

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

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_{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_{good} and N_{pop} is called the **crossover rate** (X_{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:

$$C_l = cost_l - cost_{N_{\text{good}}+1} \quad .$$

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

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

If there is large spread in costs between the 1^{st} and $N_{\text{good}}+1^{\text{th}}$ 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 μ. 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

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

    $\boxed{\text{REPEAT NumRuns TIMES:}}$
    ∗ Step 5. Set the seed value for the random number generator.

        $\boxed{\text{REPEAT UNTIL STOPPING CRITERION IS SATISFIED}}$
        (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 $\underline{\text{pairing}}$ procedure based on the current population
            and the cumulative pairing probabilities.
        • Step 7. Perform the $\underline{\text{mutation}}$ (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 (`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:

LP4	LP relaxation model: size $O(n^4)$ (scales: 0.50,0.65,0.75,0.85)
LP3	LP relaxation model: size $O(n^3)$ (scales: 0.50,0.65)

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" mutation rate was a cause of early terminations at local optima. The mutation rates of 0.02 and 0.01 turned out to be the most successful ones, balancing between early convergence and divergence.

Table 1. Optimal ($n = 10, 15$) and best known b&b solutions ($n = 20, 25$).

	$n = 10$		$n = 15$	
T	IP	`mvs:1`	IP	`frv:1`
5K	588,146,366.08*	22s**	2,077,497,554.40*	2m54s**
10K	590,319,444.91*	15s**	2,081,771,855.96*	17m31s**
15K	591,925,150.34*	26s**	2,085,013,594.34*	1h04m23s**
20K	595,312,146.30*	45s**	2,090,585,171.66*	1h24m00s**
25K	599,792,785.02*	1m14s**	2,097,082,629.72*	2h10m17s**
30K	601,657,025.49*	41s**	2,102,282,543.87*	50m27s**
35K	603,378,402.99*	40s**	2,108,901,922.12*	3h44m14s**
40K	604,183,537.20*	31s**	2,115,825,446.99*	11h59m46s**
45K	605,647,773.32*	31s**	2,119,614,638.52*	1h39m19s**
50K	608,410,251.54*	1m03s**	2,122,598,385.21*	42m02s**
55K	610,373,726.22*	24s**	2,124,475,670.93*	19m20s**
60K	611,440,169.01*	33s**	2,126,567,996.70*	7m55s**
65K	612,564,897.21*	50s**	2,127,114,506.79*	7m29s**
70K	612,801,746.26*	34s**	2,128,870,166.59*	14m02s**
75K	613,049,425.70*	32s**	2,130,478,028.17*	7m21s**
80K	613,523,033.22*	39s**	2,134,187,613.32*	14m18s**
85K	613,718,181.16*	47s**	2,136,985,890.45*	16m11s**
90K	614,518,103.98*	40s**	2,140,590,873.08*	25m48s**

	$n = 20$		$n = 25$	
T	b&b IP	real time	b&b IP	real time
5K	4,760,597,761.12*	12h53m53s**	7,493,751,307.07	38m16s
10K	4,765,328,011.28	>10d	7,517,302,371.68	37m07s
15K	4,768,548,150.98	20m25s	7,538,439,959.13	39m56s
20K	4,775,110,172.59	30m07s	7,554,812,107.27	39m14s
25K	4,780,578,087.91	1h54m31s	7,589,114,393.45	40m54s
30K	4,786,079,729.43	2h01m59s	7,618,611,025.87	40m23s
35K	4,791,843,336.38	1h54m39s	7,636,093,898.62	41m25s
40K	4,797,828,016.39	51m27s	7,649,452,061.29	39m46s
45K	4,800,278,554.55	2h29m00s	7,673,051,979.15	43m27s
50K	4,805,485,611.14	2h59m45s	7,679,528,793.88	39m03s
55K	4,813,282,996.80	3h36m21s	7,689,012,147.62	41m20s
60K	4,819,579,985.91	3h37m52s	7,691,787,901.56	41m20s
65K	4,820,999,167.72	4h13m18s	7,695,053,844.32	41m20s
70K	4,832,724,329.68	4h54m15s	7,725,519,740.46	48m45s
75K	4,833,741,457.95	3h13m11s	7,732,884,371.03	45m01s
80K	4,833,758,187.42	3h03m00s	7,737,209,844.71	41m31s
85K	4,834,616,306.27	19h32m20s	7,737,620,000.80	41m31s
90K	4,837,651,132.57*	2d22h53m13s**	7,741,542,134.50	47m29s

b&b	: branch and bound
IP	: optimal objective value to the integer program (Podnar et al. [10])
b&b IP	: best known objective value to the integer program obtained by b&b (Podnar et al. [10])
`mvs:1`	: times from the b&b `mvs` run (rule=1) (Podnar et al. [10])
`frv:1`	: times from the b&b `frv` run (rule=1) (Podnar et al. [10])
*	: IP known to be optimal
**	: time needed to prove optimality
T	: threshold for discounting

The population size should not be too large, because of the expensive fitness evaluations. However, the size should not be too small either, limiting the search space. 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.

Table 2. Preliminary Results: Genetic Algorithm Strategy; size $O(n^4)$

$\mu = 0.10$ $N_{\text{init}} = 32$ $N_{\text{pop}} = 16$ $N_{\text{good}} = 8$ $N_{\text{bad}} = 8$ MaxGen=20

	$n = 10$			$n = 15$			$n = 20$		
	all	1st	gap IP	all	1st	gap IP	all	1st	gap b&b IP
T	time	time	`mvs:1`	time	time	`frv:1`	time	time	gap LP
	0	0	-	20	0	-	20	0	* 0.00001%
5K	19s	19s	22s	5m59s	1m17s	2m54s	31m27s	11m06s	0.00035%
	0	0	-	20	0	0.00019%	20	0	0.00071%
10K	14s	14s	15s	7m53s	1m39s	17m31s	35m40s	11m59s	0.00516%
	0	0	-	20	0	0.00022%	20	0	0.00075%
15K	21s	21s	26s	9m29s	1m58s	1h04m23s	38m15s	13m28s	0.00921%
	20	0	-	20	0	0.05431%	20	0	-0.00555%
20K	2m35s	29s	45s	13m21s	3m06s	1h24m00s	49m22s	18m20s	0.05013%
	20	0	-	20	0	0.19030%	20	0	-0.01807%
25K	2m35s	28s	1m14s	17m12s	4m56s	2h10m17s	53m08s	20m25s	0.05260%
	0	0	-	20	0	0.12177%	20	0	-0.00305%
30K	30s	30s	41s	16m40s	5m36s	50m27s	1h02m04s	25m59s	0.03218%
	20	0	0.03324%	20	0	0.14495%	20	0	0.01670%
35K	1m58s	36s	40s	19m29s	7m12s	3h44m14s	1h05m25s	28m23s	0.09199%
	0	0	-	20	0	0.05529%	20	0	-0.00861%
40K	28s	28s	31s	21m55s	7m38s	11h59m46s	1h18m06s	36m32s	0.10020%
	0	0	-	20	0	0.15335%	20	0	-0.01296%
45K	24s	24s	31s	25m17s	7m30s	1h39m19s	1h15m21s	35m21s	0.05187%
	0	0	-	20	0	-	20	0	0.11506%
50K	28s	28s	1m03s	19m39s	6m03s	1h42m02s	1h42m14s	50m17s	0.19067%
	0	0	-	20	0	-	20	0	-0.00497%
55K	25s	25s	24s	15m57s	5m18s	19m20s	1h49m39s	52m11s	0.12107%
	0	0	-	20	0	0.01877%	20	0	-0.01147%
60K	29s	29s	33s	10m32s	4m29s	7m55s	1h44m18s	50m36s	0.14685%
	2	0	-	20	0	0.12124%	20	0	0.14446%
65K	34s	29s	50s	12m46s	4m42s	7m29s	2h07m20s	1h01m45s	0.23551%
	0	0	-	20	0	-	20	0	0.01942%
70K	30s	30s	34s	12m43s	5m04s	14m02s	2h27m09s	1h15m02s	0.26491%
	0	0	-	20	0	-	20	0	-0.02000%
75K	30s	30s	32s	11m18s	4m49s	7m21s	2h06m31s	1h15m33s	0.17762%
	0	0	-	20	0	-	20	0	0.20240%
80K	29s	29s	39s	12m17s	5m12s	14m18s	2h11m49s	1h16m02s	0.31105%
	0	0	0.20749%	20	0	-	20	0	0.20747%
85K	30s	30s	47s	11m06s	4m31s	16m11s	2h27m49s	1h20m36s	0.24768%
	2	0	0.21285%	20	0	0.00100%	20	0	* -
90K	35s	29s	40s	19m11s	6m18s	25m48s	1h48m37s	1h15m26s	0.03356%

b&b: branch and bound

all: number of generations in the run

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

gap IP: $\frac{\text{obj}-\text{IP}}{\text{IP}}$; obj = best (GA) solution; IP = optimal objective value to the integer program (Podnar et al. [10])

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

gap LP: $\frac{\text{obj}-\text{LP}}{\text{LP}}$; LP = optimal objective value to the LP relaxation (Podnar et al. [10])

$-$: no gap (obj=IP)

`mvs:1`: times from the b&b `mvs` run (rule=1) (Podnar et al. [10])

`frv:1`: times from the b&b `frv` run (rule=1) (Podnar et al. [10])

$*$: IP known to be optimal

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_{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 `frv:1` from Podnar et al. [10] were gathered for the comparison. The `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 $O(n^4)$

$\mu = 0.02$ $N_{\text{init}} = 30$ $N_{\text{pop}} = 25$ $N_{\text{good}} = 10$ $N_{\text{bad}} = 15$ MaxGen=15

	$n = 15$			$n = 20$			$n = 25$		
T	all time	1st time	gap IP `frv:1`	all time	1st time	gap b&b IP gap LP	all time	1st time	gap b&b IP gap LP
5K	15 6m18s	0 1m28s	0.00053% 2m54s	15 24m59s	0 11m31s	* - 0.00034%	15 3h36m43s	0 52m50s	0.05655% 0.09671%
10K	15 8m51s	0 2m03s	0.00225% 17m31s	15 32m39s	0 12m35s	0.00040% 0.00485%	15 4h11m04s	4 1h45m55s	-0.04423% 0.31015%
15K	15 11m23s	0 2m24s	0.01450% 1h04m23s	15 47m03s	0 13m55s	0.00000% 0.00846%	15 4h34m10s	4 1h51m15s	-0.06275% 0.57357%
20K	15 14m40s	0 3m47s	0.01153% 1h24m00s	15 54m45s	0 20m27s	0.00000% 0.05568%	15 5h29m34s	1 1h14m57s	-0.00722% 0.84800%
25K	15 15m26s	10 11m31s	0.00735% 2h10m17s	15 56m51s	0 21m21s	-0.02845% 0.04220%	15 7h56m29s	1 1h19m30s	-0.07801% 1.23418%
30K	15 17m57s	5 9m31s	0.03276% 50m27s	15 1h09m34s	6 42m43s	0.02414% 0.05939%	15 7h59m07s	2 1h30m50s	-0.20166% 1.50189%
35K	15 18m54s	11 15m51s	0.08574% 3h44m14s	15 1h10m56s	5 40m57s	-0.01631% 0.05895%	15 6h15m55s	6 2h45m34s	-0.34085% 1.59292%
40K	15 22m24s	5 12m50s	0.09369% 11h59m46s	15 1h34m35s	13 1h27m58s	-0.04705% 0.06171%	15 11h09m18s	6 2h43m23s	-0.18185% 1.93391%
45K	15 22m14s	14 21m35s	0.11335% 1h39m19s	15 1h17m29s	4 47m36s	-0.03676% 0.02805%	15 11h02m06s	3 1h54m56s	-0.33135% 2.09435%
50K	2 7m21s	0 6m24s	- 42m02s	15 1h48m37s	4 1h07m07s	-0.01925% 0.05627%	15 23h37m41s	0 1h06m15s	0.00000% 2.52023%
55K	4 7m11s	0 5m13s	- 19m20s	15 1h48m35s	3 1h04m07s	-0.01508% 0.11094%	15 23h34m10s	2 1h40m42s	0.04424% 2.69224%
60K	4 6m18s	0 4m48s	- 7m55s	14 1h28m25s	10 1h21m17s	-0.09672% 0.06148%	>13 >17h43m	2 1h40m45s	0.17819% 2.86686%
65K	5 7m06s	2 5m59s	- 7m29s	11 1h51m58s	8 1h43m40s	-0.02853% 0.06237%	n/a n/a	n/a n/a	n/a n/a
70K	4 7m36s	2 7m00s	- 14m02s	15 2h23m40s	8 1h52m01s	-0.09341% 0.15180%	n/a n/a	n/a n/a	n/a n/a
75K	4 7m57s	0 5m24s	- 7m21s	15 2h04m03s	4 1h34m33s	-0.13411% 0.06329%	n/a n/a	n/a n/a	n/a n/a
80K	5 9m14s	2 8m08s	0.03028% 14m18s	15 2h12m40s	15 2h12m40s	-0.02978% 0.07861%	n/a n/a	n/a n/a	n/a n/a
85K	12 13m43s	0 5m18s	- 16m11s	8 1h48m36s	2 1h33m51s	0.08471% 0.12488%	n/a n/a	n/a n/a	n/a n/a
90K	2 9m05s	0 7m43s	0.00100% 25m48s	8 1h41m56s	2 1h28m22s	* - 0.03356%	n/a n/a	n/a n/a	n/a n/a

b&b: branch and bound

all: number of generations in the run

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

gap IP: $\frac{\text{obj}-\text{IP}}{\text{IP}}$; obj = best (GA) solution; IP = optimal objective value to the integer program (Podnar et al. [10])

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

gap LP: $\frac{\text{obj}-\text{LP}}{\text{LP}}$; LP = optimal objective value to the LP relaxation (Podnar et al. [10])

$-$: no gap (obj=IP)

`frv`:1: times from the b&b `frv` run (rule=1) (Podnar et al. [10])

$*$: IP known to be optimal

n/a: computation stopped due to extensive time consumption

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 early termination of the algorithm.

Table 4. Genetic Algorithm Strategy; size $O(n^3)$

$\boxed{n = 25}$ $N_{\text{init}} = 30$ $N_{\text{pop}} = 25$ $N_{\text{good}} = 10$ $N_{\text{bad}} = 15$ MaxGen=10

	$\mu = 0.02$			$\mu = 0.01$			$\mu = 0.005$		
	all	1^{st}	gap b&b IP	all	1^{st}	gap b&b IP	all	1^{st}	gap b&b IP
T	time	time	gap LP	time	time	gap LP	time	time	gap LP
	10	0	0.05655%	10	6	0.04480%	10	9	0.03311%
5K	2h39m26s	52m50s	0.09671%	2h23m48s	1h45m04s	0.08496%	2h17m15s	2h09m19s	0.07325%
	10	4	-0.04423%	10	10	-0.08605%	10	9	-0.13254%
10K	3h01m38s	1h45m55s	0.31015%	2h49m57s	2h49m57s	0.26818%	2h44m55s	2h34m38s	0.22153%
	10	4	-0.06275%	10	8	-0.14689%	10	9	-0.19984%
15K	3h14m39s	1h51m15s	0.57357%	2h59m09s	2h35m42s	0.48890%	2h52m25s	2h41m13s	0.43561%
	10	1	-0.00722%	10	10	-0.10317%	10	10	-0.11327%
20K	3h37m42s	1h14m57s	0.84800%	3h01m51s	3h01m51s	0.75124%	2h52m52s	2h52m52s	0.74105%
	10	1	-0.07801%	10	8	-0.31182%	10	9	-0.30274%
25K	4h26m19s	1h19m30s	1.23418%	3h17m38s	2h50m29s	0.99730%	3h11m04	2h58m31s	1.00650%
	10	2	-0.20166%	10	7	-0.34448%	10	10	-0.53589%
30K	4h01m27s	1h30m50s	1.50189%	3h13m05s	2h33m55s	1.35663%	3h11m06s	3h11m06s	1.16195%
	10	6	-0.34085%	10	10	-0.34395%	10	10	-0.44467%
35K	4h17m27s	2h45m34s	1.59292%	3h30m32s	3h30m32s	1.58976%	3h17m15s	3h17m15s	1.48709%
	10	6	-0.18185%	10	8	-0.23156%	10	10	-0.34185%
40K	4h34m49s	2h43m23s	1.93391%	3h22m41s	2h54m33s	1.88224%	3h03m21s	3h03m21s	1.76962%
	10	3	-0.33135%	10	6	-0.51303%	10	9	-0.41875%
45K	4h44m37s	1h54m56s	2.09435%	3h36m51s	2h40m48s	1.90825%	3h14m52s	3h02m16s	2.00380%
	10	0	0.00000%	10	8	-0.43734%	10	10	-0.49526%
50K	9h08m18s	1h06m15s	2.52023%	3h30m08s	3h01m11s	2.07187%	3h15m26s	3h15m26s	2.01248%
	10	2	0.04424%	10	8	-0.23817%	10	10	-0.56519%
55K	8h45m50s	1h40m42s	2.69224%	3h52m00s	3h18m15s	1.40235%	3h22m07s	3h22m07s	2.06668%
	10	2	0.17819%	10	6	-0.25637%	10	10	-0.44540%
60K	8h27m36s	1h40m45s	2.86686%	3h34m28s	2h33m56s	2.42063%	3h26m20s	3h26m20s	2.22653%
	10	0	0.00000%	10	0	0.00000%	10	10	-0.26787%
65K	9h17m31s	1h10m49s	2.72748%	3h38m48s	1h11m21s	2.72748%	3h36m35s	3h36m35s	2.45232%
	10	0	0.00000%	10	2	-0.09146%	10	10	-0.58673%
70K	7h30m57s	1h14m55s	3.13420%	3h50m47s	1h38m47s	3.03987%	3h14m40s	3h14m40s	2.52901%
	10	0	0.00000%	10	8	-0.45226%	10	10	-0.45584%
75K	8h56m55s	1h16m51s	3.23252%	3h20m15s	2h53m13s	2.76563%	3h03m49s	3h03m49s	2.76194%
	10	1	0.02855%	10	10	-0.24259%	10	9	-0.48273%
80K	9h26m57s	1h33m54s	3.31975%	3h34m29s	3h34m29s	3.03969%	3h11m35s	2h58m15s	2.79165%
	10	0	0.00000%	10	2	-0.14705%	10	9	-0.44486%
85K	10h31m05s	1h16m12s	3.29573%	3h57m29s	1h39m23s	3.14384%	3h07m45s	2h54m35s	2.83621%
	10	0	0.00000%	10	1	-0.06519%	10	9	-0.38919%
90K	12h55m46s	1h18m14s	3.34810%	4h06m41s	1h30m05s	3.28073%	3h03m45s	2h52m46s	2.94587%

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

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

	MaxGen=10		MaxGen=60	
T	time	improvement over branch and bound	time	improvement over branch and bound
10K	2.75h	0.133%	10.04h	0.207%
20K	2.88h	0.113%	13.09h	0.369%
30K	3.19h	0.536%	13.07h	0.826%
40K	3.06h	0.342%	12.00h	0.863%
50K	3.28h	0.495%	15.01h	1.004%
60K	3.44h	0.445%	16.75h	0.967%
70K	3.24h	0.587%	15.08h	0.843%
80K	3.19h	0.483%	9.21h	0.636%
90K	3.06h	0.389%	7.01h	0.710%

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 $O(n^3)$

$\boxed{n = 25}$ $\mu = 0.005$ $N_{\text{init}} = 30$ $N_{\text{pop}} = 25$ $N_{\text{good}} = 10$ $N_{\text{bad}} = 15$ MaxGen=60

T	all	time	1^{st}	time	obj	gap b&b IP	gap LP
5K	50	7h11m33s	41	6h16m30s	7,493,719,289.43	-0.00043%	0.03971%
10K	52	10h02m36s	43	8h19m48s	7,501,767,383.89	-0.20666%	0.14715%
15K	49	10h24m29s	40	8h25m32s	7,512,960,034.28	-0.33800%	0.29657%
20K	60	13h05m34s	60	13h05m34s	7,526,910,253.30	-0.36933%	0.48280%
25K	60	14h02m48s	60	14h02m48s	7,535,000,011.50	-0.71305%	0.59080%
30K	53	13h04m10s	44	10h15m08s	7,555,677,024.14	-0.82606%	0.86683%
35K	60	15h10m30s	60	15h10m30s	7,566,292,197.33	-0.91410%	1.00854%
40K	47	11h59m52s	38	9h16m01s	7,583,474,002.89	-0.86252%	1.23792%
45K	55	19h26m26s	46	12h34m27s	7,589,534,633.00	-1.08845%	1.31882%
50K	48	15h00m48s	39	10h48m26s	7,602,461,197.92	-1.00355%	1.49139%
55K	51	19h06m42s	42	12h17m08s	7,614,835,764.07	-0.96471%	1.65659%
60K	48	16h44m46s	39	10h39m25s	7,617,375,060.38	-0.96743%	1.69049%
65K	48	17h04m11s	39	11h42m24s	7,639,488,742.41	-0.72209%	1.98570%
70K	51	15h04m59s	42	11h04m17s	7,660,380,154.29	-0.84317%	2.26460%
75K	34	12h47m43s	25	7h24m02s	7,681,756,626.93	-0.66117%	2.54997%
80K	29	9h12m30s	20	5h34m01s	7,687,965,101.07	-0.63647%	2.63285%
85K	51	17h33m36s	42	12h07m40s	7,665,993,622.92	-0.92569%	2.33954%
90K	27	7h00m43s	18	4h47m53s	7,686,551,105.87	-0.71033%	2.61397%

b&b: branch and bound

all: number of generations in the run

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

gap LP: $\frac{\text{obj}-\text{LP}}{\text{LP}}$; obj = best (GA) solution; LP = optimal objective value to the LP relaxation (Podnar et al. [10])

gap b&b IP: $\frac{\text{obj}-\text{ b\&b IP}}{b\&b\ IP}$; 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: $\mu = 0.005$ $N_{\text{init}} = 30$ $N_{\text{pop}} = 25$ $N_{\text{good}} = 10$ $N_{\text{bad}} = 15$. 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. $\frac{1}{1+x}$), 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 $\mathbf{T} = T_{km}$ 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

1. S. Abdinnour-Helm. A hybrid heuristic for the uncapacitated hub location problem. *European Journal of OR*, 106:489–499, 1998.
2. S. Abdinnour-Helm and M. A. Venkataramanan. Solution approaches to hub location problems. *Annals of Operations Research*, 78:31–50, 1998.
3. J. F. Campbell. Hub location and the p-hub median problem. *Operations Research*, 44(6):923–935, 1996.
4. S. Deris S, S. Omatu, H. Ohta, Lt. Cdr S. Kutar and P. A. Samat. Ship maintenance scheduling by genetic algorithm and constrained-based reasoning. *European Journal of OR*, 112:489–502, 1999.

5. A. Ernst and M. Krishnamoorthy. Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. *European Journal of OR*, 104:100–112, 1998.
6. J. G. Klincewicz. Avoiding local optima in the p-hub location problem using tabu search and GRASP. *Annals of Operations Research*, 40:282–302, 1992.
7. Z. Michalewicz. *Genetic algorithms + Data structures = Evolution programs.* Springer Verlag, New York, 1994.
8. M. O'Kelly and D. Bryan. Hub location with flow economies of scale. *Transportation Research -B*, 32(8):605–616, 1998.
9. H. Pirkul and E. Rolland. New heuristic solution procedures for the uniform graph partitioning problem: Extensions and evaluation. *Computers and Operations Research*, 21:895–907, 1994.
10. H. Podnar, J. Skorin-Kapov and D. Skorin-Kapov Network cost minimization using threshold based discounting. *European Journal of OR*, 137(2):371–386, 2002.
11. D. Skorin-Kapov, J. Skorin-Kapov and M. E. O'Kelly. Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal of OR*, 94:582–593, 1996.
12. D. Whitley, T. Starkweather and D. Shaner. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. in Davis L. (Ed.), *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, 1991.
13. G. Zhou and M. Gen. A note on genetic algorithms for degree-constrained spanning tree problems. *Networks*, 30:91–95, 1997.
14. G. Zhou and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of OR*, 114:141–152, 1999.