A Genetic Algorithm for the Uncapacitated Network Design Problem

Jozef Kratica[1], Dušan Tošić[2], Vladimir Filipović[2] and Ivana Ljubić[3]

[1]Serbian Academy of Sciences and Arts, Institute of Mathematics, Strumicka 92/5, 11 000 Belgrade, Serbia, Yugoslavia, E-mail: jkratica@matf.bg.ac.yu, WWW: http://www.geocities.com/~jkratica
[2]University of Belgrade, Faculty of Mathematics, Studentski trg 16, Belgrade, Serbia, Yugoslavia, E-mail: {dtosic | vladaf} @matf.bg.ac.yu, WWW: http://www.matf.bg.ac.yu/~vladaf
[3]Vienna Univesity of Technology, Institute for Computer Graphics, Favoritensstasse 9, Vienna, Austria, E-mail: ivana@ads.tuwien.ac.at, WWW: http://www.ads.tuwien.ac.at/people/Ljubic.html

Abstract. In this paper is presented genetic algorithm (GA) for solving the uncapacitated network design problem (UNDP) with single sources and destinations for each commodity. UNDP is base in class of the network design problems, but it is still NP-hard. Robust GA implementation is additionally improved by caching GA technique. Computational results on instances up to 10 commodities, 30 nodes and 120 edges are reported.

1. Introduction

1.1. Problem formulation

Let be C set of commodities, G = <N, E> oriented graph where N is set of nodes and E is set of edges. For every commodity $k \in C$ is given $o(k)$, $d(k)$, and r(k) as source, destination and quantity of shipment. If we want to use some edge $(i,j) \in A$ fixed cost f_{ij} must be paid beforehand, and transportation cost c_{ij}^k for every commodity which is transported by that edge.

In this problem we find transportation plan for all commodities with minimal overall cost that contains fixed and transportation costs. In this problem edges have not capacity limits, which means every commodity can be transported by same path. In that situation, without loosing generality, all transportation costs c_{ij}^k may be multiplied by r_k and quantities r_k are normalized to 1.

Fact that this problem is NP-hard is proved in [6] by showing that the Steiner tree problem (a well known NP-hard problem) is a special case of UNDP.

Problem can be mathematically formulated as:

$$min\left(\sum_{k\in C}\sum_{(i,j)\in A}c_{ij}{}^{k}\cdot x_{ij}{}^{k}+\sum_{(i,j)\in A}f_{ij}\cdot y_{ij}\right) \tag{1}$$

with constraints

$$\sum_{j:(i,j)\in A}x_{ij}{}^{k} - \sum_{i:(j,i)\in A}x_{ji}{}^{k}=b_i^k \qquad \forall i\in N,\ \forall k\in C \tag{2}$$

$$0\le x_{ij}^{k}\le y_{ij} \qquad \forall(\mathrm{i,j})\in A,\ \forall \mathrm{k}\in\ C \tag{3}$$

$$y_{ij}\in\{0,1\} \qquad \forall(\mathrm{i,j})\in\ E \tag{4}$$

In formulas (1) - (4) x_{ij}^{k} is quantity of commodity k which is transported by edge (i,j)$\in E$. Binary variables y_{ij} indicates if edge (i,j) $\in E$ is established (y_{ij}=1) or not (y_{ij}=0). Induced subgraph that contains all established edges denote by G' = <N,E'> , E' = { (i,j) | y_{ij} = 1 }.

Flow of commodity k throw node i is denoted by $\mathrm{b_i}^k$ as be seen from formula (5).

$$\mathrm{y_{ij}}=\begin{cases}1, & edge\ (i,j)\ is\ established\\ 0, & edge\ (i,j)\ is\ not\ established\end{cases} \qquad \mathrm{b_i}^k=\begin{cases}1, & i=o(k)\\ -1, & i=d(k)\\ 0, & otherwise\end{cases} \tag{5}$$

This problem can be formulated also as problem for improving design of existing network. It can be done by setting zero to fixed costs f_{ij} of all established edges, which automatically setting this edges as established (y_{ij} = 1) in newly obtained solution. This guarantee appearance of all previously established edges in improved network design.

1.2. Related work

Several different methods were used for solving this problem:

A Lagrangean heuristic (that is described in [5]) uses a Lagrangean relaxation as subproblem. It solved the Lagrange dual with subgradient optimization combined with a primal heuristic yielding primal feasible solutions. In [1] was also described Lagrangean relaxation for solving the UNDP (and several similar applications) but otherwise than previous, relaxed was coupling to the fixed charges.

Benders decomposition was used for solving UNDP in [10], which effectively use the fact of large number of continuous variables and relatively small number of integer variables.

Paper [7] presents a Lagrangean heuristic within branch-and-bound framework as a method for finding the exact optimal solution of the UNDP. This approach efficiently combines previously described methods (Lagrangean relaxation and Benders decomposition).

An efficient dual ascent approach was proposed in [2]. The primal feasible solutions were found using the primal local search heuristic. Method is shown to very quickly achieve very good solutions (1-4% of duality gap).

In [11] was given good survey of other methods used for solving UNDP. It also provides a unifying view for synthesizing many network design models and proposes a unifying framework for deriving many network design algorithms.

1.3. Genetic algorithms

Under the most frequent classification, genetic algorithms and some related techniques, together with Fuzzy logic and Neural networks, are part of so called Soft computing.

According to Darwin, individuals in population are competing for resources. The facts that lie in essence of natural selection process are:

- Better fitted individuals more often survive and they have stronger influence on forming new generations;
- Individual in new generation is formed by recombination of parent's genetic material;
- From time to time mutation (random change of genetic material) takes place.

Basic steps in GA are:

1. **Initialization** – generating initial population by random sampling.
2. **Evaluation** – calculating fitness for all items in population.
3. **Selection** – choosing surviving items in population, according to values of fitness function.
4. **Recombination** (includes crossover and mutation) – changing item's representation.
5. **Repeating steps 1.-4.** until fulfilling finishing criteria.

2. The GA implementation

2.1. Encoding and objective value function

UNDP formulation given in (1) - (5) have $(|C|+1) \cdot |A|$ variables and in that form is not suitable for solving by GA on larger instances. In our approach only variables y_{ij} are binary encoded in genetic code. If y_{ij} are fixed, x_{ij}^k values may be obtained by shortest path algorithm performed on G'. In this approach genetic code is slightly smaller and contains only $|A|$ bits, but on other hand, shortest path algorithm must be performed for every commodity $k \in C$.

2.2. Genetic operators

Fine grained tournament selection is applied with average number of competitors f_{tur} = 5.6. Theoretical aspects of this selection method can be found in [3], and results of applying it on some real-world problems in [4] Uniform crossover operator is performed with probabilities p_{cross} = 0.85 and p_{unif} = 0.3. Simple mutation with probability given in (6) is chosen.

$$p_{mut} = \frac{1}{2 \cdot |A|} \tag{6}$$

2.3. Initialization and generation replacement strategy

Initial population is randomly generated by setting each bit (y_{ij}) independently with probability 3/4 to 1. In this way chances that induced subgraph G' is connected is high, which provide that almost all solutions are feasible.

In our experiments population contains 150 individuals and is performed steady-state generation replacement with elitist strategy. In every generation, the best 2/3 of population (100 individuals) goes directly into the next generation and only the worst 1/3 of population is replaced by offsprings. That approach preserves good individuals and prevents their loss by unlucky use of some genetic operator.

2.4. Caching the GA

Afterwards, run-time performance of the GA is improved by caching technique. The idea of caching is used to avoid permanent attempts to compute the same objective value ([8]). Instead of that, objective values are remembered and reused. It is especially important when that computing is time expensive. If the value of an individual has to be computed, and it is already cached, we just read it from the cache.

In this paper a simple but efficient Least Recently Used (LRU) strategy for caching is used. It is implemented by a hash-queue data structure, which saves the individuals and the corresponding values. The queue size is a parameter that depends on memory size and other performance constraints. More information about caching GA technique can be found in [8] and [9].

3. Computational results

3.1. Computer environment and problem instances

GA is implemented by a program written in ANSI C. The whole implementation is portable for various computer platforms (MS-DOS, MS Windows, UNIX, ...). The program is divided into two parts:

1. Kernel of GA, containing common GA functions applicable for various problems;
2. Specific GA functions for SPLP, related to: I/O operations, initialization and objective value function.

Authors produced randomly generated groups of UNDP instances MA-MJ. Every group contains 5 different instances generated by different random seed (100, 200, 300, 400, and 500). Input parameters for this generator are integers |C|, |N|, |A| and real numbers c_{min}, c_{max}, f_{min}, f_{max}. Particular values of those parameters are given in the Table 2.

The members of transportation cost matrix are randomly generated from interval [c_{min}, c_{max}]. After that, the sum $S_{ij} = \sum_{k \in C} c_{ij}^{k}$ is computed for every edge (i, j). The sum S_{ij} denotes a cumulative value of all transportation costs on particular edge. In the final phase, an inverse scaling into the interval [f_{min}, f_{max}] is performed by the following formula:

$$f_i = f_{\max} - \frac{\left(S_i - S_{\min}\right)\cdot\left(f_{\max} - f_{\min}\right)}{S_{\max} - S_{\min}} \tag{7}$$

This corresponds to some real situations, where smaller transportation costs c_{ij} produce higher fixed costs f_{ij} and vice versa.

Table 1. Properties of generated UNDP instances.

Instance group	Dimension C, N, A	f	c	File size
MA	10×5×15	2-6	1-3	2.5 KB
MB	10×10×32	2-6	1-3	5 KB
MC	10×15×50	2-6	1-3	7.7 KB
MD	10×20×70	2-6	1-3	10.7 KB
ME	10×30×120	2-6	1-3	18.2 KB
MF	50×30×120	2-6	1-3	72 KB
MG	10×50×250	2-6	1-3	37 KB
MH	50×50×250	2-6	1-3	148 KB
MI	10×100×700	2-6	1-3	103 KB
MJ	50×100×700	2-6	1-3	412 KB

3.2. Experimental results

All of our experiments were done running on IBM-PC compatible computers with an AMD Pentium III processor at 750 MHz, and 1 GB of RAM. Since genetic operators: selection, crossover and mutation are undeterministic, every problem instance is executed 20 times (except MJ group).

Since no specific implementation that solves UNDP is publicly available, GA is compared to general mixed-integer programming (MIP) codes. Instances are converted to the MIP format and programs OSL (by IBM) and LP_SOLVE are used in order to solve it. Optimal solutions are not obtained for instances in groups MF, MH and MJ (except for the simplest instance MF1 that is solved by OSL for 5 hours and 41 minutes).

Table 2 and Table 3. contain obtained results. Those tables. have following columns:

- Name of UNDP instance;
- Optimal solution obtained by MIP code;
- Average number of generations necessary for finishing GA execution;
- Average run-time in seconds;
- Best GA solution.

Table 2. Obtained results for groups MA-ME.

UNDP Instance	Optimal solution	Avg. gener.	Avg. run-time (s)	Best solution by GA
MA1	55.193	31.9	0.020	optimal
MA2	53.221	32.5	0.020	optimal
MA3	57.684	36.9	0.020	optimal
MA4	56.047	38.6	0.020	optimal
MA5	55.215	34.8	0.020	optimal
MB1	81.755	104.8	0.102	optimal
MB2	85.945	78.0	0.082	optimal
MB3	81.445	84.5	0.089	optimal
MB4	94.471	94.6	0.097	optimal
MB5	90.216	102.0	0.095	optimal
MC1	87.974	305.1	0.321	optimal
MC2	107.735	262.1	0.293	optimal
MC3	85.750	233.3	0.237	optimal
MC4	87.560	252.9	0.279	optimal
MC5	110.498	248.9	0.284	optimal
MD1	105.877	1056	1.041	optimal
MD2	133.368	915.8	0.859	optimal
MD3	114.454	896.9	0.907	optimal
MD4	105.776	920.3	0.926	optimal
MD5	116.365	829.8	0.843	optimal
ME1	137.823	3418	4.532	optimal
ME2	129.454	4377	5.601	optimal
ME3	127.715	4182	4.946	optimal
ME4	135.951	4206	4.258	optimal
ME5	140.869	5506	6.457	optimal

Table 3. Obtained results for groups MF-MJ.

UNDP Instance	Optimal solution	Avg. gener.	Avg. run-time (s)	Best solution by GA
MF1	704.810	5288	54.400	optimal
MF2	-	4305	47.146	713.520
MF3	-	5792	58.901	720.708
MF4	-	5943	67.383	671.535
MF5	-	5527	57.136	741.679
MG1	111.160	6848	12.031	optimal
MG2	144.824	7135	16.386	145.658
MG3	129.799	6743	11.668	132.167
MG4	126.965	6833	14.500	optimal
MG5	132.544	6808	12.947	135.200
MH1	-	12428	358.154	885.605
MH2	-	10724	307.772	878.704
MH3	-	13916	401.364	886.907
MH4	-	12715	379.558	875.131
MH5	-	12838	337.036	848.895
MI1	125.352	13072	58.729	128.972
MI2	117.254	11512	53.040	118.451
MI3	136.258	12596	61.498	143.132
MI4	147.563	12959	66.826	157.165
MI5	136.423	13443	60.518	155.468
MJ1	-	25801	2414.86	1248.595
MJ2	-	13544	1214.35	1252.019
MJ3	-	26730	1917.38	1164.988
MJ4	-	32524	2480.68	1136.297
MJ5	-	38286	3261.73	1192.727

4. Conclusion and future work

In this paper is proposed a new GA-based approach for solving an uncapacitated network design problem. Binary encoding, efficient objective value function, rank-based selection, steady-state generation replacement with elitist strategy and caching GA technique are especially convenient in this case. Using those techniques our GA approach solved UNDP instances up to 50 nodes, 100 edges and 700 commodities in reasonable time. For almost all instances where optimal solution is known, GA reaches that solution. Authors are expecting that for instances where optimal solution is not determined by other methods, solution obtained by GA have same quality.

There are many directions for further investigations: testing our approach for larger size problem instances, hybridizing of GA with other methods for solving UNDP, testing parallel GA implementation on this problem and applying this approach to some other similar problems.

In summary, we feel that the experimental results reported here are encouraging, and that the future work in this direction should yield new insights into the field of network design.

References

[1] Ahuja R.K., Magnanti T.L, Orlin J.B. (1993) Network Flows, Theory, Algorithms and Applications, Prentice Hall Int.

[2] Balakrishnan, A. Magnanti, T.L, Wong R.T. (1989) A Dual-Ascent Procedure for Large-Scale Uncapacitated Network Design, Operations Research **37**, 716-740

[3] Filipović V. (1998) Proposition for improvement tournament selection operator in genetic algorithms, MS thesis, University of Belgrade, Faculty of Mathematics (in Serbian)

[4] Filipović V, Kratica J, Tošić D, Ljubić I. (2000) Fine Grained Tournament Selection for the Simple Plant Location Problem, Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications - WSC5, 152-158

[5] Hellstrand J, Holmberg K. (1994) A Langrangean Heuristic Applied to the Uncapacitated Network Design Problem, Technical Report LiTH-MAT/OPT-WP-1994-04, Department of Mathematics, Linköping Institute of Technology, Sweden

[6] Holmberg K, Jörnsten K, Migdalas A. (1986) Decomposition methods applied to discrete network design, Technical Report LiTH-MAT/OPT-WP-1986-07, Department of Mathematics, Linköping Institute of Technology, Sweden

[7] Holmberg K, Hellstrand J. (1998) Solving the uncapacitated network design problem by a Langrangean heuristic and branch-and-bound, Operations Research 46, 247-259

[8] Kratica J. (1999) Improving Performances of the Genetic Algorithm by Caching, Computers and Artificial Intelligence **18** (3), 271-283

[9] Kratica J. (2000) Parallelization of Genetic Algorithms for Solving Some NP-Complete Problems, Ph.D. thesis, University of Belgrade, Faculty of Mathematics (in Serbian)

[10] Magnanti T.L, Mireault P. and Wong R.T. (1986) Tailoring Benders Decomposition for Uncapacitated Network Design, Mathematical Programming Study **29**, 464-484

[11] Magnanti T.L, Wong R.T. (1984) Network Design and Transportation Planning: Models and Algorithms, Transportation Science **18**, 1-55