

UNIVERSITY OF THESSALY
SCHOOLS OF POLYTECHNICS
DEPARTMENT OF MECHANICAL ENGINEERING

Master's Thesis

**Multi Depot Vehicle Routing for inspection of Electric Grid substations
using MDVRP formulation and a hybrid genetic algorithm**

by

Dr. Spathis Vasileios

Electrical Engineer, Aristotles University of Thessaloniki

Submitted to fulfill part of
requirements for the acquisition of
Masters Degree

2015

Approved by the three members of the Commission of Inquiry:

Chief Examiner (Supervisor)	Dr Georgios Saharidis Assistant Professor in Operations Research Methods in Industrial Management Department of Mechanical Engineering, University of Thessaly
Second Examiner	Dr. George Liberopoulos Professor in Stochastic Methods in Production Management Department of Mechanical Engineering, University of Thessaly
Third Examiner	Dr. George Kozanidis Assistant Professor in Optimization Methods of Production/Service Systems Department of Mechanical Engineering, University of Thessaly

Acknowledgements

This project would not have been possible without the support of many people. Many thanks to my adviser, G. Saharidis, who read my numerous ideas and helped, make some sense of the confusion. Thanks to the Independent Power Transmission Operator for awarding me a full scholarship for attending this Master's Degree. Special thanks to my family and especially my wife for providing the necessary time and space for completing this dissertation. Special thanks to my parents, to my father and mother in law for their support and to numerous friends who endured this long process with me, always offering support and love.

Vasileios Spathis

Multi Depot Vehicle Routing for inspection of Electric Grid substations using MDVRP formulation and a hybrid genetic algorithm

VASILEIOS SPATHIS

University of Thessaly, Department of Mechanical Engineering, 2015

Supervisor: Dr. George Saharidis, Assistant Professor in Operations Research Methods in Industrial Management, Department of Mechanical Engineering, University of Thessaly

Abstract

The inspection of electric substations that support the electric grid, are a major part of the stability of the electric transmission system. This thesis addresses the routing of such inspections as an MDVRP problem. The proposed formulation is solved using optimization software. A hybrid genetic algorithm is also proposed that uses as initial population a set of feasible solution extracted from the optimization software. The proposed hybrid genetic algorithm performs well and managed almost at every scenario of initial population, to further optimize the objective produced from optimization software.

Table of Contents

1.	Literature Review	7
1.1.	Introduction	7
1.2.	MDVRP	7
1.3.	Research on VRP and MDVRP	8
1.4.	Genetic Algorithms	9
1.4.1.	Crossover	10
1.4.2.	Mutation	11
1.5.	Conclusion	13
2.	Problem Description - Inspection of substations.....	14
2.1.	Introduction	14
2.1.1.	General description of the Energy backbone	14
2.1.2.	Importance of substations	15
2.1.3.	Inspection	15
2.2.	Conclusion	16
3.	Problem formulation	17
3.1.	Introduction	17
3.2.	Indices and sets of the problem.....	17
3.3.	Parameters.....	17
3.4.	Variables	17
3.5.	Conclusion.....	20
4.	Methods used for solution	21
4.1.	Introduction	21
4.2.	Running standard optimization in CPLEX.....	21
4.3.	The Genetic Algorithm	24
4.3.1.	The aim of the genetic algorithm	27
4.4.	The data used from CPLEX and Genetic Algorithm	27
4.5.	Conclusion.....	28
5.	Results.....	29
5.1.	Introduction	29
5.2.	Steps for solving and comparing the results.....	29
5.3.	Memory consumed for CPLEX optimization	29
5.4.	Gap convergence	30
5.5.	Objective convergence.....	30
5.6.	Results after 24hours	31
5.7.	Results after 1,5 hours	32

5.8.	Results after 5min	32
5.9.	Comparison of Objectives	33
6.	Discussion	35
6.1.	Further development Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.	
7.	Bibliography	36
8.	Appendix	39
8.1.	The Google Matrix Code	39
8.2.	Code to refine the distance matrix produced from google maps	42
8.3.	The mathematical model as coded for CPLEX (in C++)	44
8.4.	The distance file used:	54
8.5.	The chromosomes exported from CPLEX optimization	55
8.6.	The genetic algorithm (C++ program)	55
8.7.	CPLEX output.....	67
8.8.	Generation of scenarios	69
8.9.	Scenarios	71

List of Figures

Figure 1: Multi depots and customers to be served.....	7
Figure 2: A feasible routing solution.....	8
Figure 3: A transmission line	14
Figure 4: A substation	15
Figure 5: The substations of Central Greece	15
Figure 6: The calculation of cost.....	23
Figure 7: A feasible route	23
Figure 8: The genetic algorithm.....	24
Figure 9: Optimization memory consumption	29
Figure 10: The reduction of Gap during optimization	30
Figure 11: Objective convergence for various scenarios.....	31
Figure 12: CPLEX and Genetic Algorithm results after 24h of optimization	31
Figure 13: CPLEX and Genetic Algorithm results after 1.5h of optimization	32
Figure 14: CPLEX and Genetic Algorithm results after 5min of optimization	33
Figure 15: The results for each scenario.....	33

List of Tables

Table 1: Sets of the problem	17
Table 2: Indices used to formulate the problem	17
Table 3: The variables of the problem.....	18

1. Literature Review

1.1. Introduction

A brief literature review is presented on optimization for vehicle route problems and multi depot vehicle route problems. Genetic algorithms regarding those problems will be discussed and certain aspects of the genetic evolution are displayed.

1.2. MDVRP

Many distribution systems widely depend on the routing and scheduling of vehicles through a set of customers requiring service. The Vehicle Routing Problem (VRP) is the designing of a group of minimum-cost vehicle routes, originating and terminating at a central depot, for a fleet of vehicles that services a set of customers with known demands. Each customer is serviced exactly once and, furthermore, all customers must be assigned to vehicles without exceeding vehicle capacities [1].

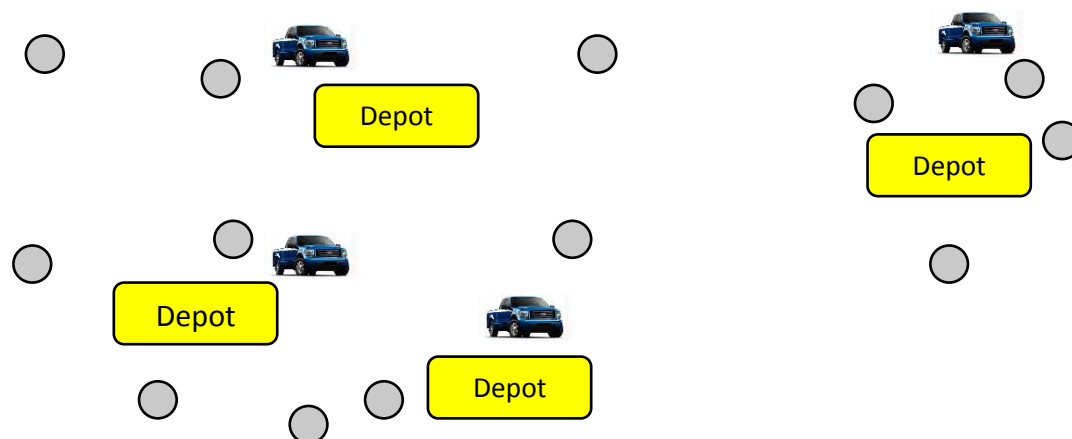


Figure 1: Multi depots and customers to be served

MDVRP (Multi Depot Vehicle Route Problem) on the other hand introduces several depots that are ready to serve customers, with only one to be assigned for every customer. Furthermore the rest of the restrictions of the general VRP problem are valid. Therefore, each customer is visited (serviced) once, each vehicle must leave and return to the same depot and the fleet size at each depot is specified. As expected, the objective function is on both occasions directly related to the total trip time, or distance, or fuel consumption of the set of trips that cover customer need.

The MDVRP is NP-hard [1] [2] therefore analytic solution is only available for a small set of customers and depots. The development of heuristic algorithms for this problem class is of primary interest.

1.3. Research on VRP and MDVRP

There has been a generous list of papers introducing several aspects on VRP and MDVRP. Several papers introduce extensions on VRP such as time windows, backhauls, pickup and delivery VRP etc.

Exact algorithms were proven to be time consuming and hardly ever applicable Liong et al. [3]. Therefore, heuristics are usually more applicable on realistic problems.

The oldest heuristic on MDVRP was developed by Tillman and Cain [4] and Wren and Holliday [5]. Tillman and Cain introduced a heuristic based on the saving algorithm of Clark and Wright [6] which implies that every combination of routes is resulting to cost saving. In this heuristic, as many other heuristic developed at this early stage, the improvement stage of the initial solution was missing resulting to a poor optimization overall. Whereas the latter group (Wren and Holliday) introduced a heuristic of two stages in which a feasible solution is obtained in the first stage and an improvement of this feasible solution is calculated in the second stage. This procedure of obtaining an initial solution first and then improving it is one of the two common structures of the heuristics developed to solve the MDVRP [7].

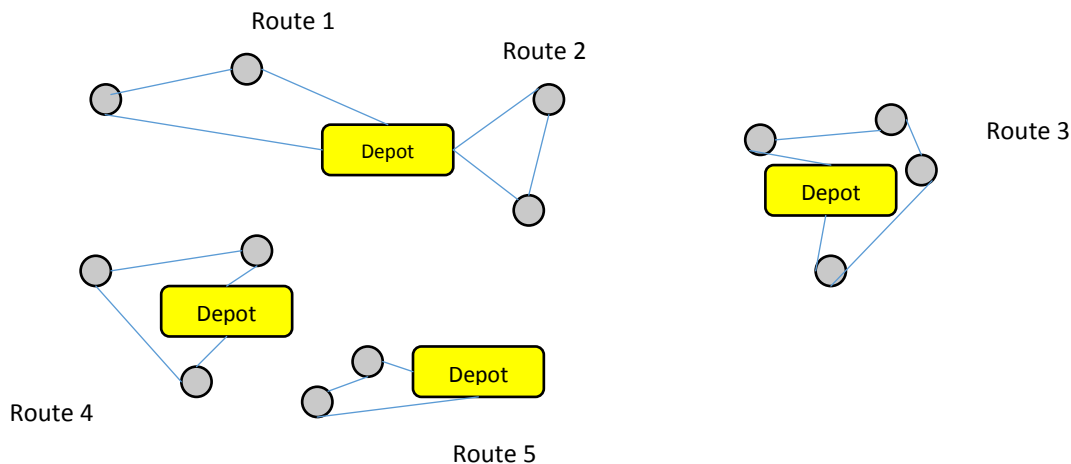


Figure 2: A feasible routing solution

The second well known structure involves decomposing the MDVRP into sub problems first and then solving these sub problems separately before connecting them iteratively. Golden et al. [8] in 1977 developed an algorithm using the cluster first - route second principle. This algorithm clusters customers in the first stage according to the distances to the depots closest and second closest to the customers. A few years later Raft [9] developed an algorithm for which the problem is decomposed in five, instead of two, stages. The five stages are solved separately and then connected iteratively.

Chao et al. [10] applied the cluster first - route second principle presented by Golden et al. [11] to obtain an initial solution and improve this solution by changing the depot assignments of the customers. Hence, they combine the two commonly applied structures in heuristics solving the MDVRP. In the improvement procedure, Chao et al. [10] applied the record-to-record approach [12].

Tabu search algorithms on the MDVRP have been first introduced by Renaud et al. [13] and Cordeau et al. [14]. These algorithms use memory structures to memorize the examined

solutions. A tabu search algorithm will not examine a solution more than once as it marks the visited solutions as tabu in its memory for a certain number of iterations [15].

The years after 2000, research has primarily been dedicated to extensions of the MDVRP but the classical MDVRP has been investigated as well. Pisinger and Ropke [16] developed a unified heuristic to solve five extensions of the classical VRP including the MDVRP. This heuristic uses the adaptive large neighborhood search framework designed by (Ropke and Pisinger, 2006 [17]). This framework expands and contracts the search for a better solution by choosing adaptively among a number of heuristics that are able to insert and remove customers in a route.

Ho et al. [7] were the first to propose two hybrid genetic algorithms to solve the MDVRP. The first algorithm generates the initial population randomly while the second algorithm uses the savings algorithm and the nearest neighbor heuristic. The nearest neighbor heuristic is a routing heuristic that solves the TSP. It searches for the nearest unvisited customer location until every location is visited and then returns to the starting location. Mirabi et al. [18] presented three hybrid heuristics to solve the MDVRP where these heuristics combine components from constructive heuristic search and improvement methods. With constructive heuristic search a complete solution, i.e. all customers satisfied, is 'constructed' by extending an empty solution.

Vidal et al. [19] introduced another heuristic which is a hybrid genetic algorithm. This so called Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) addresses three extensions of the VRP including the MDVRP. Vidal et al. apply the giant-tour representation and the Split algorithm as Prins [20] did to obtain a routing scheme corresponding to the randomly determined assignment scheme. The giant-tour representation implies that from each depot one routes starts, serving all customers assigned to that depot and then returns to the depot. The Split algorithm is used to find the optimal segmentation of the giant-tour into separate routes. One of the differences between regular genetic algorithms and the algorithm developed by Vidal et al. is the control on the diversity of the population. Vidal et al. do not include a mutation phase, instead they apply "advanced population diversity management schemes" of including feasible as well as infeasible solutions and refreshing the population when a pre-specified maximum number of iterations made without improving the best solution obtained so far is reached. Also, a diversity contribution is included in the fitness measure used to evaluate the chromosomes. This enhances the evolution of the population while avoiding it to converge prematurely.

Another difference is the inclusion of an education phase to further enhance the improvement of chromosomes. In this phase local-search procedures educate the chromosomes by improving their routing schemes.

Vidal et al. demonstrated that their algorithm is the best performing method to approach the MDVRP.

1.4. Genetic Algorithms

Genetic algorithms (GA) is a subset of evolutionary algorithms and are adaptive, heuristic and mimic natural evolution described by Darwin. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate

solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

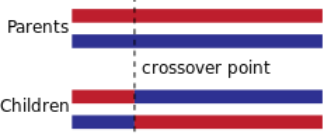
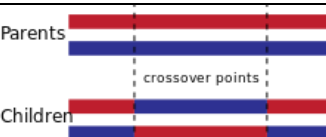
The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. [21]

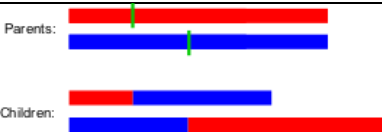


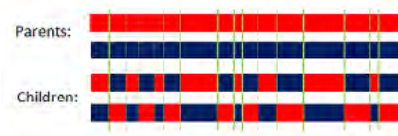
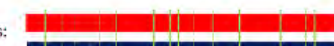

A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming. Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

1.4.1. Crossover

Crossover as mentioned above is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Cross over is a process of taking more than one parent solutions and producing a child solution from them. There are methods for selection of the chromosomes which include the Fitness proportionate selection, also known as fitness proportionate selection where the probability of an individual to be selected is proportional of the fitness value, the Boltzmann selection, the Tournament selection, the Rank selection, the Steady state selection and the Local selection method.

One Point Crossover		A single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms.
Two Point Crossover		Two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms

Cut and Splice	 <p>Parents: </p> <p>Children: </p>	Another crossover variant, the "cut and splice" approach, results in a change in length of the children strings. The reason for this difference is that each parent string has a separate choice of crossover point.
Uniform Crossover and half Uniform Crossover	 <p>Parents: </p> <p>Children: </p>	The Uniform Crossover uses a fixed mixing ratio between two parents. Unlike one- and two-point crossover, the Uniform Crossover enables the parent chromosomes to contribute the gene level rather than the segment level.

After a selection method is applied the selected chromosomes are combined in order to produce their children. Combination methods include One point crossover, two point crossover and cut and splice crossover as described on the table below:

A special category of crossover techniques apply to ordered chromosomes, when the representation implies a certain order at the chromosome and therefore the children produced need to apply to the same rules.

One such case is when the chromosome is an ordered list, such as an ordered list of the cities to be travelled for the traveling salesman problem. There are many crossover methods for ordered chromosomes. However, sometimes a crossover of chromosomes produces conjunctions which violate the constraint of ordering and thus need to be repaired.

1. Partially matched crossover (PMX): In this method, two crossover points are selected at random and PMX proceeds by position wise exchanges. The two crossover points give matching selection. It affects cross by position-by-position exchange operations. In this method parents are mapped to each other, hence we can also call it partially mapped crossover [22]
2. Cycle crossover (CX): Beginning at any gene i in parent 1, the i -th gene in parent 2 becomes replaced by it. The same is repeated for the displaced gene until the gene which is equal to the first inserted gene becomes replaced (cycle).
3. Order crossover operator (OX1): A portion of one parent is mapped to a portion of the other parent. From the replaced portion on, the rest is filled up by the remaining genes, where already present genes are omitted and the order is preserved.
4. order-based crossover operator (OX2)
5. position-based crossover operator (POS)
6. voting recombination crossover operator (VR)
7. alternating-position crossover operator (AP)
8. sequential constructive crossover operator (SCX)[6]

1.4.2. Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation occurs during evolution according to a user-definable mutation

probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are inversion and floating point mutation. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions and scrambles.

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random (or semi-random) selection with a weighting toward those that are fitter. [23]

Different mutation types are introduced:

- Bit string mutation

The mutation of bit strings ensue through bit flips at random positions.

Example:

1 0 1 0 0 1 0

↓

1 0 1 0 1 1 0

The probability of a mutation of a bit is $1/\lambda$, where λ is the length of the binary vector. Thus, a mutation rate of 1 per mutation and individual selected for mutation is reached.

- Flip Bit

This mutation operator takes the chosen genome and inverts the bits (i.e. if the genome bit is 1, it is changed to 0 and vice versa).

- Boundary

This mutation operator replaces the genome with either lower or upper bound randomly. This can be used for integer and float genes.

- Non-Uniform

The probability that amount of mutation will go to 0 with the next generation is increased by using non-uniform mutation operator. It keeps the population from stagnating in the early stages of the evolution. It tunes solution in later stages of evolution. This mutation operator can only be used for integer and float genes.

- Uniform

This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

- Gaussian

This operator adds a unit Gaussian distributed random value to the chosen gene. If it falls outside of the user-specified lower or upper bounds for that gene, the new gene value is clipped. This mutation operator can only be used for integer and float genes.

1.5. Conclusion

A brief literature review on research papers regarding the VRP and MDVRP problems were presented. Also the genetic algorithms on vehicle routing problems were discussed. In particular the different processes in a genetic algorithm were introduced and small examples were given. The next Chapter will discuss the problem that is challenged and the parameters that define it.

2. Problem Description - Inspection of substations

2.1. Introduction

In this Chapter we will discuss the importance of the good working conditions of the transmission grid of a country and the substations between. A working real model of such a system is the set of substations that cover central Greece. This system is presented and the different aspects of the human inspections are introduced.

2.1.1. General description of the Energy backbone

Customers of electric companies worldwide are convinced that electricity is there every second of the day ready to be used. What is not generally known is that electricity is actually made available as soon as someone chooses to consume. Electric power that runs through the transmission lines is produced for every customer at the moment he switches on an electric appliance.

This small detail of the electric energy makes the reliable transmission of power a very important stage of the chain that connects production and consumption. Transmission of energy must be available 24/7 and cannot be interrupted at any point. If an interruption is to be made or happens accidentally (for example a failure of a transmission line) then an adequate number of alternative transmission routes must be available. Lack of backup routes is a major reason for a massive power cut.

Electric-power transmission is the bulk transfer of electrical energy, from generating power plants to electrical substations located near demand centers. This is distinct from the local wiring between high-voltage substations and customers, which is typically referred to as electric power distribution. Transmission lines, when interconnected with each other, become transmission networks. The combined transmission and distribution network is known as the "power grid".

A transmission line as the one shown on the right connects to substations. A substation is a part of an electrical generation, transmission, and distribution system. Inside a substations voltage is transformed from higher level to lower level, or the reverse, or several other important functions regarding the quality of the voltage and the protection of the system are performed. Between the generating station and consumer, electric power may flow through several substations at different voltage levels.



Figure 3: A transmission line

Generally, substations are unattended, relying on SCADA for remote supervision and control. Through the SCADA the control center of the transmission company can monitor several aspects of the process, and can interfere if necessary to recover part of a failed circuit.

A substation may include transformers to change voltage levels between high transmission voltages and lower distribution voltages, or at the interconnection of two different transmission voltages. The word substation comes from the days before the distribution system became a grid. As central generation stations became larger, smaller generating plants were converted to distribution stations, receiving their energy supply from a larger plant instead of using their own

generators. The first substations were connected to only one power station, where the generators were housed, and were subsidiaries of that power station.

2.1.2. Importance of substations

As mentioned above. Substations are unattended and rely on remote supervision and control software. Nevertheless, the human inspection of the substation is been considered as a major part of the maintenance process.

Substations are considered as the interconnection part of the transmission lines grid (the ultra-high and high voltage grid) and the distribution grid (the middle and low voltage grid). The transformation of power to lower or higher voltage and the automated control and protection processes realized in a substation must be also 24/7 available.



Figure 4: A substation

2.1.3. Inspection

Human inspection of this kind of infrastructure is essential part of the process. Every transmission company that operates transmission lines grid is using human inspection for maintenance purposes. Highly trained inspectors visit substations every a few days (regularly twice a month is considered as a good balance of cost and effectiveness) and run a certain list of routine checks in the substation. For instance a simple walk nearby the transformers in order to hear the noise or to observe leaks, would provide important information that cannot be read to any characteristic obtained through the SCADA software until maybe it's too late.

Regularly the results of such an inspection is been handed to the technical personnel who will take immediate actions such as an extraordinary service or replacement of fault equipment, or rescheduling of the planned service of equipment involved.

The above action will actually prevent a major fault to the system and help to preserve the availability of the electric power as promised by the electric company to the final customer.

A regular human inspection of every substation is a time consuming process due to the placement of the substations. The substations are sporadically built in a large area. Typically substations are 10 to 50 km from each other, and an area of 15.000 km² would have 30 to 50 substations. Greece for example has 331 substations that cover an area of 131.000 km². A map of substations that support Central Greece is shown on the right.

One the other hand the inspectors are usually stationed in certain locations. For example, for the stations of Central Greece, inspectors are located at 4 spots, considered as "depots". Every route of inspection is commencing at a depot, runs through several substations and is finished at the same depot.

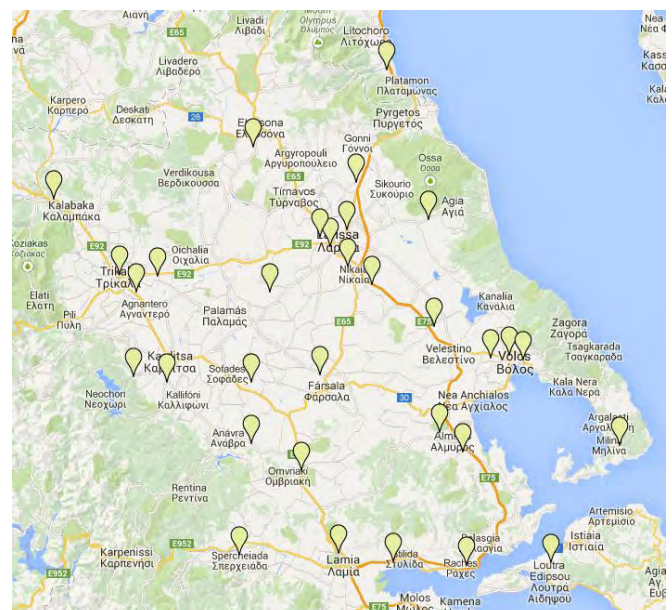


Figure 5: The substations of Central Greece

At each substation the inspector is spending an amount of time usually depending on the size of the substation and the irregularity of the equipment stationed at the site. This amount of time can be estimated for every substation beforehand.

Furthermore, each route of inspection cannot exceed a certain time. It has been proven from experience, and it can be assumed as common sense that after a few inspections on the same day, the inspector is “losing interest” due to fatigue and routine.

Lastly, the inspection needed of the next month is determined at the last week of the running month. Although that someone could assume that the visits are definite, the experience has proven otherwise. For example, the inspection of a substation at the first two weeks of December can be excluded if a major part of the substation will be visited by technical teams for a scheduled maintenance. At that day several experts will visit the substation and will be able to inspect thoroughly the equipment and the substation field; therefore the regular inspection can be excluded from the scheduling. While this seems like a little gain for the inspection program, if we take into account that every two weeks 5% to 10% of the substations will be visited by maintenance teams, we can imagine that the scheduled inspection program can be dramatically reduced.

This thesis will address the problem of inspection of substation using a general multi depot vehicle routing algorithm and hybrid genetic algorithms. At the final stage a path of solutions will be suggested. The algorithm will be tested at several scenarios of substations.

2.2. Conclusion

At this Chapter the problem of the human inspection of the scattered electric substations is presented. This is a MDVRP problem with many particularities. The importance of the uninterrupted energy transfer and transformation was also emphasized. Lastly, parameters of the solutions and certain logical restriction were introduced. At the next Chapter the mathematical formulation of the problem is presented.

3. Problem formulation

3.1. Introduction

The mathematical formulation used to address this MDVRP problem depends on the work of Dondo et al. [24] regarding a time window multi depot vehicle routing problem. This generic formulation was adapted to the inspection routing problem. The formulation is presented and explained thoroughly.

3.2. Indices and sets of the problem

The sets created and the corresponding indices used for the formulation are displayed at the Table 1 on the right.

The “cost” travelling from a substation or depot to another substation or depot is considered not the km between them but the time needed for travelling. This was

s,k	Substations
d	Depots
t	Days for routing

TABLE 2: INDICES USED TO FORMULATE THE PROBLEM

$S=\{1,2,...,n\}$	Substations
$D=\{1,2,...,m\}$	Depots
$A(s,k)$	Time between s and k, where $s,k \in \{S \cap D\}$
$T=\{1,2,...,d\}$	Days for inspection

TABLE 2: SETS OF THE PROBLEM

chosen because of another aspect of the problem which is the time to be consumed at each substation, since these two parameters will determine the total cost.

3.3. Parameters

The known data of the problem are:

- c_{sk} Time to travel between s and k substation (or depot) in minutes
- m_s Time that is spend to inspect the s substation in minutes
- MT Maximum time of a route (including travelling and inspecting time)

3.4. Variables

At the Table 3 below there is a list of the variables used. The cost of every route is built as we visit the next substation.

Variable	Type of Variable	Variable descriptions
CV_s	Integer	Cumulative cost of a route until s substation
X_{st}	Binary	1 if s substation is visited at t day, 0

		otherwise
Y_{sd}	Binary	1 if s substation is visited from a route beginning at d depot, 0 otherwise
R_{sk}	Binary	1 if s substation is visited before k substation, 0 otherwise
CVT_{dt}	Integer	Total cumulative cost of t day route beginning from d depot

TABLE 3: THE VARIABLES OF THE PROBLEM

The last variable makes the objective function very simple and straightforward. We want to minimize the sum of all the routes that serve the substations that take place any of the t days and start from any of the d depots. Therefore:

$$\min \sum_{d,t} CVT_{dt}$$

Subject to:

$$1. \sum_{d \in D} Y_{sd} = 1, \forall s \in S$$

$$2. \sum_{t \in T} X_{st} = 1, \forall s \in S$$

$$3. CV_s \geq (c_{ds} + m_s)(X_{st} + Y_{sd} - 1), \forall s \in S, d \in D, t \in T$$

$$4. CV_k \geq CV_s + (c_{sk} + m_k) - M(1 - R_{sk}) - M(2 - X_{st} - X_{kt})$$

$$5. CV_s \geq CV_k + (c_{sk} + m_s) - MR_{sk} - M(2 - X_{st} - X_{kt}), \forall s, k \in S, t \in T, s < k$$

$$6. CVT_{dt} \geq CV_s + c_{sd} - M(2 - X_{st} - Y_{sd}), \forall s \in S, d \in D, t \in T$$

$$7. CVT_{dt} \leq MT, \forall d \in D, t \in T$$

where M is a big number. The restrictions above make determine the feasibility of the solution as follows:

1. Each substation is assigned to one and only depot. This restriction arises from the description of the problem since we want the inspector to depart from one depot visit a number of substations and return to the same depot. We describe this restriction as a summation of all y_{sd} binary variables for all d depots for each s substation to be equal to 1. This restriction will be expanded to $k=\max(s)$ restrictions (as many as the substations).
2. Each substation is served (visited) only once. This restriction is also defined by the problem description. The restriction is a summation of all x_{st} binary variables for all t days (t can be also assumed as routes of inspections) for each s substation to be also equal to 1. Therefore for each substation s only one x_{st} will be equal to 1. This restriction will also be
3. The accumulated time of a route until the s substation is the sum of the time to travel from a depot to the substation s (parameter: c_{sk}) and the time needed to inspect the substation (parameter: m_s), taking in advance that s is the first substation in the route. This restriction applies only for the first substations for every route and can provide $s \times d \times t$ restrictions. Most of them will be neglected as only one substation for every route will be the first.
4. The accumulated time of a route when k substation is visited is the sum of the accumulated time until s substation (variable CV_s), the time travelling from s to k substation (parameter: c_{sk}) and the time to inspect k substation (parameter: m_k). All these are taking into consideration if s substation is visited before k and if s and k substation are visited the same date (t). This restriction will be expanded to $s \times d \times t$ restrictions.
5. This restriction is complementary to the last one with respect of k been visited before s .
6. The total accumulated time of the route is when the route is arriving at depot, and therefore the accumulated time until the s substation (variable: CV_s) is summed to the time needed to travel from s substation to d depot. This is controlled by the binary variables determining if s is visited at t day and if s substation is assigned to d depot. This restriction is expanded to $s \times d \times t$ restrictions
7. This restricts the total amount of traveling time of every route. The restriction is expanded to d formulas.

The formulation at Dondo paper was applied here, eliminating the time window restrictions but using the time indices as routing indices. Therefore from every depot we can have several routes, one per day. This can also be used in order to restrict the number of routes but it was irrelevant to the solution needed and therefore it was ignored. The total number of restriction is added to:

$$4(s \cdot d \cdot t) + 2 \cdot s + d$$

Therefore for a standard problem of 24 substations, 4 depots and 12 days to visit we could have 4660 restriction.

3.5. Conclusion

The mathematical formulation of the MDVRP problem regarding the inspection of the electric substations was presented. The formulation is a customization of Dondo work on time window MDVRP problem. Next Chapter the methods used to solve the problem are introduced.

4. Methods used for solution

4.1. Introduction

The MDVRP problem of the inspection of electric grid substations was solved using two different methods presented in this Chapter. Firstly the mathematical formulation was written in C++ and fed to CPLEX optimization studio. Since the problem is NP-hard the solution beyond a threshold is not expected at logical time limitations.

Furthermore a genetic algorithm also presented below, was written in C++ and run in order to find feasible solutions. The genetic algorithm is doing all the basic processes adjusted to a MDVRP problem.

4.2. Running standard optimization in CPLEX

The ILOG CPLEX Optimization Studio was used for the calculation of the analytic solutions. The code producing the input to the CPLEX studio was written in C++ and is included at Appendix 8.3. The code uses two input files for the data. The first file contains the distance matrix of every depot and substation to each other as shown at Appendix 8.4. The instance shown there is the first scenario that was optimized at CPLEX and hybrid genetic algorithm, and is also the real scenario of the substation at Central Greece. All other scenarios are a random simulation of points. The second file is a file that we determine the amount of time the inspector will spent at each substation.

The parameters and limitations that were used for the simulation were 24 substation, 4 depots and 360min max route time. These parameters and limitations were introduced due to the fact that they are the actual facts of the problem regarding the inspection of substations of Central Greece. Execution of the Code with CPLEX return the output attached at Appendix 8.7.

The solutions were saved as a list of the variables of the problem, and the real routes can be extracted from those values. For example a feasible solution can be extracted from the variables shown below:

Solution 180 with objective 2050

CVTdt(0,5)=328

CVTdt(0,7)=359

CVTdt(1,0)=239

CVTdt(1,4)=268

CVTdt(2,1)=251

CVTdt(2,6)=280

CVTdt(3,8)=325

At this part of the solution the total objective is shown as well as the total cost of each route from every deposit. For instance an inspection route proposed from deposit 0 as route 5 is 328 minutes long.

Xst(0,5)=1

Xst(1,5)=1

Xst(2,7)=1

Xst(3,7)=1

Xst(4,5)=1

Xst(5,5)=1

Xst(6,7)=1

Xst(7,4)=1

Xst(8,0)=1

Xst(9,4)=1

The x_{st} variable if 1, defines that s substation will be visited at t route. For instance substation 0 will be visited at 5th route, as well as substation 1, 4, 5. This information alone is not conclusive of the exact route proposed.

$X_{st}(10,0)=1$
 $X_{st}(11,4)=1$
 $X_{st}(12,4)=1$
 $X_{st}(13,1)=1$
 $X_{st}(14,1)=1$
 $X_{st}(15,1)=1$
 $X_{st}(16,6)=1$
 $X_{st}(17,6)=1$
 $X_{st}(18,8)=1$
 $X_{st}(19,8)=1$
 $X_{st}(20,8)=1$
 $X_{st}(21,8)=1$
 $X_{st}(22,7)=1$
 $X_{st}(23,6)=1$

$Y_{sd}(0,0)=1$
 $Y_{sd}(1,0)=1$
 $Y_{sd}(2,0)=1$
 $Y_{sd}(3,0)=1$
 $Y_{sd}(4,0)=1$
 $Y_{sd}(5,0)=1$
 $Y_{sd}(6,0)=1$
 $Y_{sd}(7,1)=1$
 $Y_{sd}(8,1)=1$
 $Y_{sd}(9,1)=1$
 $Y_{sd}(10,1)=1$
 $Y_{sd}(11,1)=1$
 $Y_{sd}(12,1)=1$
 $Y_{sd}(13,2)=1$
 $Y_{sd}(14,2)=1$
 $Y_{sd}(15,2)=1$
 $Y_{sd}(16,2)=1$
 $Y_{sd}(17,2)=1$
 $Y_{sd}(18,2)=1$
 $Y_{sd}(19,3)=1$
 $Y_{sd}(20,3)=1$
 $Y_{sd}(21,3)=1$
 $Y_{sd}(22,0)=1$
 $Y_{sd}(23,2)=1$

The y_{sd} variable if 1, determines that s substation is assigned to d depot. Therefore substation 0,1,2,3,4,5,6,22 are assigned at depot 0. Taking into account that 0,1,4,5 substations are visited at 5th route, we can safely conclude that a route visiting this substation has been proposed by the optimization.

$CV_s(0)=143$
 $CV_s(1)=64$
 $CV_s(2)=40$
 $CV_s(3)=231$
 $CV_s(4)=190$
 $CV_s(5)=274$
 $CV_s(6)=109$
 $CV_s(7)=106$
 $CV_s(8)=59$
 $CV_s(9)=53$
 $CV_s(10)=177$
 $CV_s(11)=169$
 $CV_s(12)=227$
 $CV_s(13)=77$
 $CV_s(14)=141$
 $CV_s(15)=217$
 $CV_s(16)=69$
 $CV_s(17)=141$
 $CV_s(18)=85$
 $CV_s(19)=166$
 $CV_s(20)=239$

The accumulated cost is shown for every substation. Please note the accumulated cost for substations 0,1,4,5 highlighted. The cost is building up as the substations are visited. Therefore substation 1 must be the first visited as the cost until 1 is the lowest. Then substation 0 and 4 and lastly 5. The return to depot is not shown since the arriving at the depot builds the total accumulated cost shown above.

CVs(21)=301
CVs(22)=326
CVs(23)=225

The cost of the routes as is built is shown at the graph below:

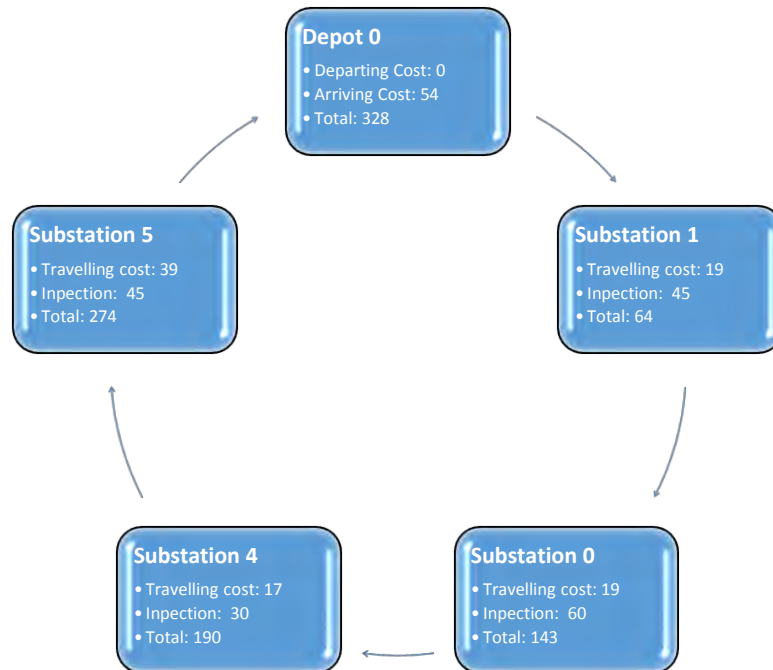


Figure 6: The calculation of cost

The departing cost from the Depot 0 is zero, and the travelling cost to the first substation of the route is 19. The inspection of the substation is 45 minutes long and therefore the accumulated cost until substation one is 64 minutes. The route is then travelling with 19min cost to substation 0, and after the inspection time of this substation (60min) the total cost is 143. Then substation 4 is visited after travelling for 17min and the accumulated cost is 190 including the inspection cost. Lastly, substation 5 is visited and then the route is ending at Depot 0 with a total cost of 328 min.

The route proposed on the map is included below:

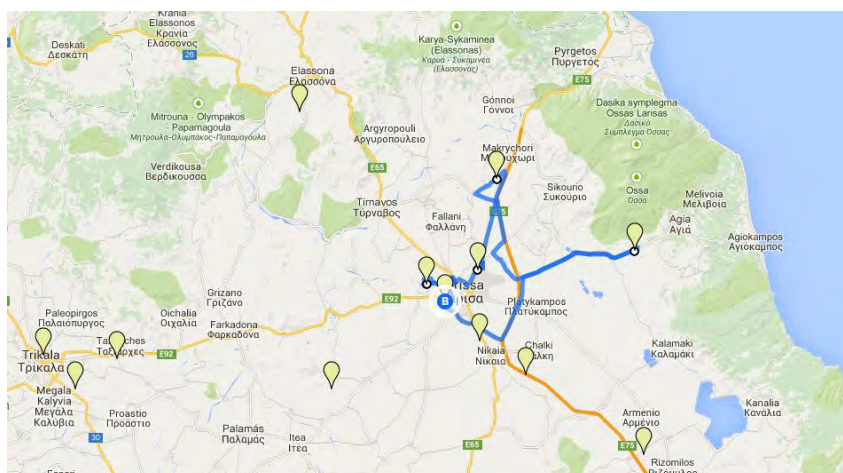


Figure 7: A feasible route

The code was run 10 times. The scenario 0 is the actual scenario with the real distances of the substations. The other 9 scenarios were built with pseudorandom process. The number of substations and depots and the restrictions of max route length (in time), were kept constant at all scenarios that were solved.

4.3. The Genetic Algorithm

A genetic algorithm was developed to be tested at the same scenarios and to check its usefulness. The code is attached at Appendix 8.6 and was written in C++.

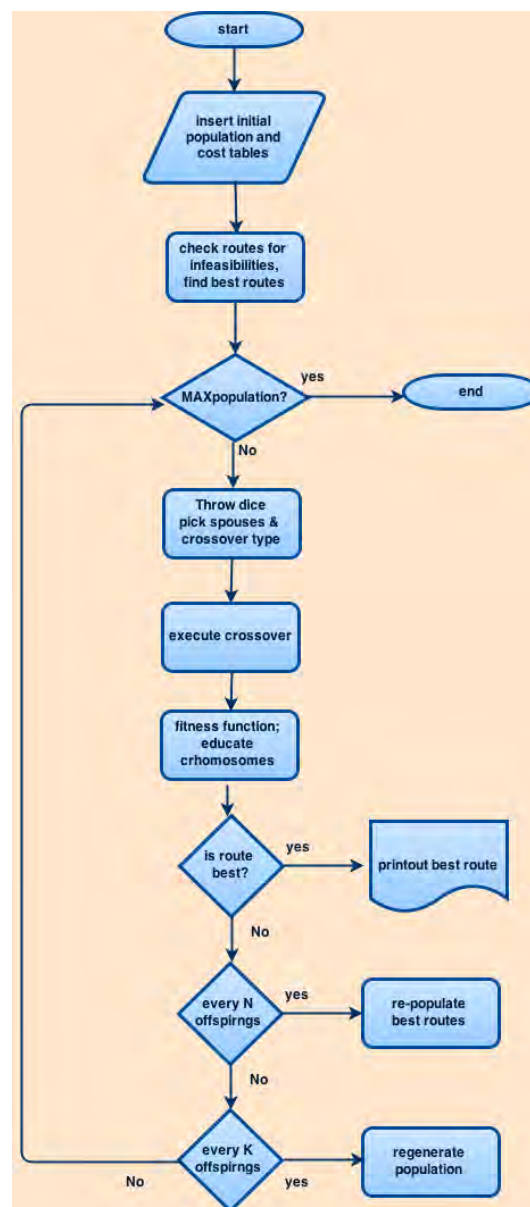


Figure 8: The genetic algorithm

The general purpose of the code was not to produce an initial population as a generic genetic algorithm. The genetic algorithm uses initial solutions as chromosomes that are introduced by the user. For this purpose for each of the produced scenarios from CPLEX the solution pool of CPLEX was used. The solution pool is a set of feasible solutions that CPLEX algorithm is finding in the way to the optimization algorithm. Those solutions were mined from CPLEX and then fed to the

genetic algorithm. The format of the chromosomes is displayed in Appendix 8.5 and sample is shown below.

```
Chromosome 1
100 0 100 100 4 5 6 100 100 1 3 22 2 100 101 9 101 101 8 101 101 7 12 11 101 101 10 101 102 13
14 15 102 102 16 17 18 23 102 103 19 20 21 103
```

The chromosome is built by routes. For the chromosome above, each three-digit of 100 or more is a depot, where each two-digit number is a substation. Each chromosome is a feasible solution. The chromosome above represents 10 routes that will visit all substations. For instance the first route will depart from depot 0 (represented with number 100) will visit substation 0 and return to depot 0.

For each chromosome that is inserted to the code or produced by the code a simple clever education is executed. This education is the simple sorting of the algorithms routes (genes) that are combined into a feasible solution in order to make it more efficient. For example for the above chromosome that is inserted at the initial population all different rerouting inside every gene are tested in order to find a near better solution. Therefore for route 100->4->5->6->100 proposed by the solution pool of CPLEX every other routing will be tested and evaluated. For every better routing than the initial a chromosome will be added at the population. Therefore route 100->5->6->5->100 and route 100->5->4->6->100 and all other routes will be evaluated in terms of minutes and if found better than the proposed then a total feasible solution (chromosome) will be created which will include the new gene and this chromosome is attached at the initial population list.

The genetic algorithm then introduces by means of randomness three different crossover-processes or a mutation process. Crossover process as introduced in genetic algorithms for VRP optimization is a rather impotent procedure. The genes that are taken from two chromosomes are combined with various ways but highly chances are that the chromosome produced will be non-feasible solution. The proof of that is shown below:

Let for instance get two chromosomes:

```
Chromosome 1
100 0 100 100 5 6 100 100 1 22 3 2 100 100 4 100 101 10 101 101 9 8 7 12 11 101 102 13 14 15
102 102 16 17 18 23 102 103 19 103 103 20 21 103

Chromosome 2
100 0 2 100 100 5 1 100 100 3 4 6 7 100 101 10 101 101 13 14 16 101 101 9 8 11 12 101 102 23 15
102 102 17 102 103 21 19 22 103 103 20 18 103
```

Now by random lets combine them in order to produce two offsprings:

```
Chromosome Offspring 1
100 0 100 100 5 6 100 100 1 22 3 2 100 100 4 100 101 10 101 101 9 8 7 12 11 101 102 23 15 102
102 17 102 103 21 19 22 103 103 20 18 103

Chromosome Offspring 2
100 0 2 100 100 5 1 100 100 3 4 6 7 100 101 10 101 101 9 8 7 12 11 101 102 13 14 15 102 102 16
17 18 23 102 103 19 103 103 20 21 103
```

From chromosome offspring one 13, 14 and 16 substations are missing while 22 is visited twice. Similarly from chromosome two, substation 22 is also missing. Both chromosomes that were produced by random crossover process are not valid solution. Several papers suggested the

“fixing” procedure of the new chromosomes by neglecting one of the duals and inserting randomly or by a clever algorithm the missing nodes.

After the insertion process of the genes and the loading of the cost tables of the specific scenario, an evaluation of all genes is introduced. The “fitness process” as called is checking every chromosome and is evaluating the objective value. This value is stored with the chromosome, as part of its id.

Then a check of infeasible solution is taken place. This was introduced in order to clarify that the initial population includes only feasible solutions. Furthermore a function is called that populates an array with the chromosomes that scored the best objective value.

Following the initial procedures of the genetic algorithm there is the reproduction loop. Two randomly chosen chromosomes either from the general population or from the population of the best chromosomes are picked. Then with an intelligent random procedure the type of the crossover is chosen. This procedure in order to randomly pick a type of crossover function examines the length of the chromosome. Beside the objective function which is an accurate representation of the quality of the solution, the length of the chromosome is a strong indication of the maturity of the feasible solution. It can be easily understood that the optimized solution will have one of the shortest length, since the nodes will be swiftly combined in order to achieve the longest feasible routes that meet the restrictions and minimize the total cost.

The longer the chromosome, the more routes are included, each one starting and ending at a depot. For example an immature solution would be to have as many routes as nodes. Every route would start from a depot, visit only one substation and finish at the depot. The chromosomes that would describe such a solution would be quite long.

Immature chromosome 100 0 100 100 1 100 100 2 100 100 3 100 100 4 100 100 5 100 e.t.c
--

At those chromosomes the genetic algorithm is working on feasible conjunctions of routes by joining two routes that belong to the same depot, in order to produce a new and hopefully better route that will cover more nodes. The routes that are produce are instantly checked for feasibility (the fitness function) and if valid, they are added to the population.

If the chromosomes are quite short then the routes that are included cover cleverly the nodes. At this point the genetic algorithm is breaking randomly routes in order to produce a more primitive feasible solution. The purpose of this action is to regenerate a simpler and less robust solution that will then be the new raw material for the genetic evolution.

The mutation process combines two different feasible solutions and randomly exchanges the position of two genes in order to produce new chromosomes. If the new chromosomes are feasible and valid then they are added to the population.

The above process can be regarded as simple and trustworthy with minimum CPU surcharge that both optimizes and regenerates the population.

The new chromosomes as well as the chromosomes of the initial population are cleverly educated in order to improve their fitness performance. This procedure is successful generates new solutions that are also added to the population.

Several other procedure were written in order to clean the population from twins, display routes for debugging purposes, etc.

Lastly a procedure that generates an array with the best feasible solution is called every K^{th} generations. This procedure checks the population and locates the m better unique solutions. This population is preferable used from the genetic algorithm as propagating material, according to a random variable uniformly distributed. Another procedure is called every M^{th} generations in order to remove chromosomes (twins, worst scored etc) and regenerate the population.

The genetic algorithm was starting with the initial population fed by the solution pool of CPLEX and run until a number of chromosomes are created. Better solutions are displayed during the running process.

4.3.1. The aim of the genetic algorithm

Since CPLEX was time consuming for the first results and endlessly running for the optimized solution, the genetic algorithm was written to try to accelerate the process. During the first tests it was understood that after the first hours of CPLEX optimization the problem was becoming memory consuming. This was slowing down the process and new better solution was hard to be found. For example as shown at Appendix 8.7, for the first scenario after 2 hours we get a solution of objective 2007 and we had to wait for another 30 hours to get the next top solution. Solution with objective 1987 was found 32.5 hours from the beginning of the optimization. This solution remained the best even after 72 hours of optimization.

The size of the problem after the first 2 hours was already 12Gbyte, and 30 hours later, when the next better solution (and final for the whole 72 hours) was found the size was 160GByte (uncompressed).

The genetic algorithm was used with the solution of the solution pool of the first two hours, when the problem is still small. Those solutions were used by the genetic algorithm in order to achieve a better solution. The process was run for all 10 scenarios.

4.4. The data used from CPLEX and Genetic Algorithm

In order to solve the problem, several data were collected from various sources. First of all a list of substations in Central Greece was written up and all the coordinates were gathered. In order to solve the problem the distance between every substation to each other as well as the depots-substations distances needed to be calculated. That was done using Google maps and the more specifically the Distance Matrix service [25]. Code was written in php and run several times in order to get all necessary data. The code is attached at Appendix 8.1.

Moreover, although the distance of the substations from each other was recorded, the useful information and hard to compute otherwise, was the time needed to travel from one substation to the other. This time was also a data mined from Google maps Matrix service. After a quick review of the distances in minutes it was made clear that a refined version of these numbers was needed. The code that “refined” the data and the details can be found at Appendix 8.2.

The above data was combined into the first scenario to be run. With a simple code another 9 pseudorandom scenarios were generated which kept certain aspects unchanged, such as the number of depots, the number of substations, the time to inspect each substation and the magnitude of the distances. The code is included in Appendix 8.8.

4.5. Conclusion

The methods to address the MDVRP problem are explained thoroughly with examples. The full code written for various aspects of the solutions is attached at Appendix. The use of CPLEX in conjunction with genetic algorithm specially designed for MDVRP is presented. This hybrid solution is introduced in order to achieve good results in more logical time windows.

5. Results

5.1. Introduction

The codes proposed at the Chapter 4 were executed for 10 scenarios of data. Each scenario is included at Appendix 8.9. In this Chapter the results of optimization of the two different methods of optimization will be presented. The discussion of the results is included at the last Chapter.

5.2. Steps for solving and comparing the results

Firstly each scenario of data describing distances between substations and depots, was optimized using CPLEX for 24 hours. From each optimization apart from the final solution, the solution pool was mined and chromosomes were produced for the genetic algorithm.

The chromosomes from the first 1.5 hours of running were fed as initial population for the genetic algorithm. The genetic algorithm was run for a target population of 100.000 (time consumed for this execution was less than 5 minutes) and the best routes proposed as well as the objective function were recorded.

The chromosomes from the total 24 hours of CPLEX optimization were also fed to the genetic algorithm. The genetic algorithm was run for a target population of 100.000 and the best routes were noted.

Lastly, CPLEX was allowed to solve the problem for only 5 minutes. All solutions in the solution pool were also fed to the genetic algorithm for optimization. The results were also recorded.

Each genetic optimization was run for more than one time, since much of the procedure is randomly oriented and the total population was restricted to a certain number.

5.3. Memory consumed for CPLEX optimization

The optimization of the problem using CPLEX had a significant impact at the memory consumption after excessive execution of the code.

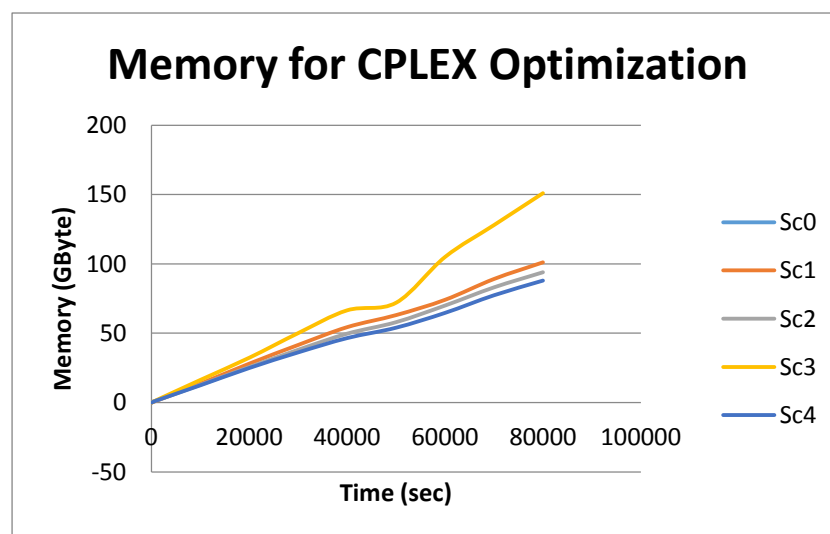


Figure 9: Optimization memory consumption

Memory consumption using CPLEX optimization software was the major crux, as expected. NP-hard problems such as the under study problem will try to use as much space as possible in order to fit all data produced from the “infinite” number of combinations that will be tried out. Figure 9 displays the memory consumption through time for each of the scenarios. As can easily be noted, five hours of optimization enlarge the problem as much as 20Gbyte. The optimization from this point and on was executing with slower pace since the reading and writing to the disk of such an enormous file size of nodes is taking a significant amount of time.

5.4. Gap convergence

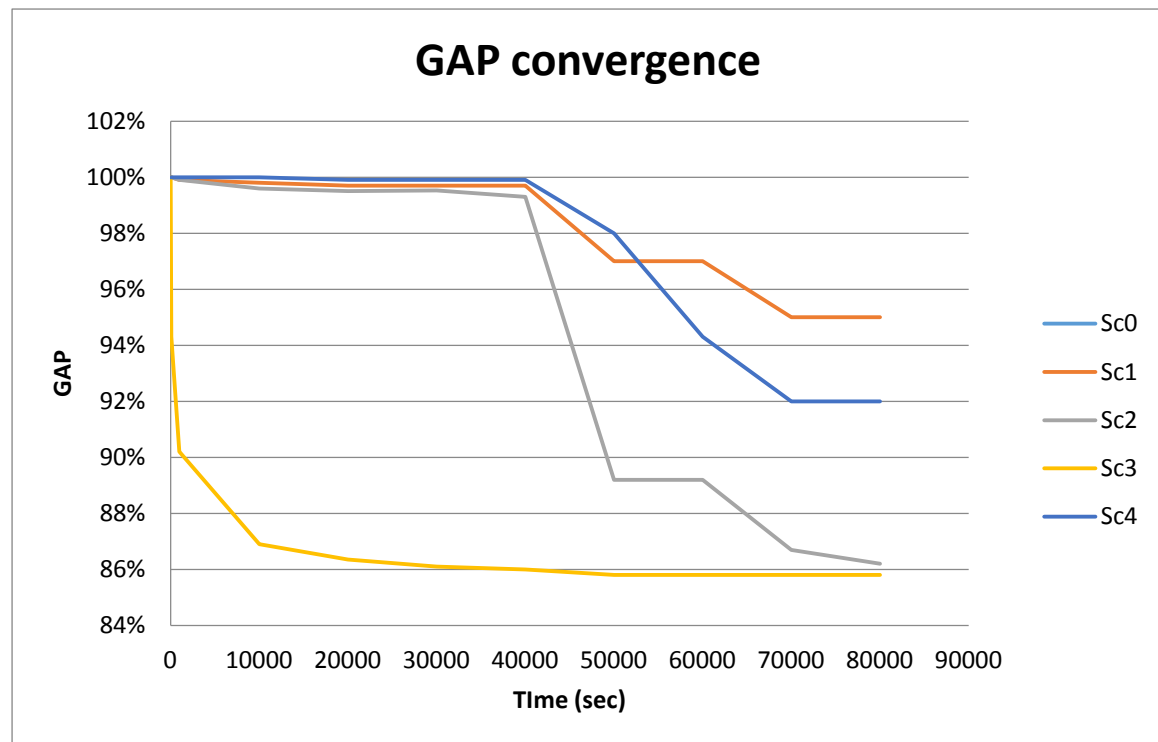


Figure 10: The reduction of Gap during optimization

At Figure 10 the gap convergence is shown against optimization time for all scenarios. The gap is reducing in slow pace and at best it was no less than 85%. Such a gap output can be explained due to the enormous size of the problem for analytic analysis. Please also note that some significant reduction from the 100% initial gap took place only after almost 12 hours of optimization for most of the scenarios.

5.5. Objective convergence

The convergence of the objective for each of the scenarios can also be explained due to the size of the problem. Figure 11 presents the real objective value against optimization time. Significant convergence at the objective value for each of the 10 scenarios is taken place at the first 6 to 10 hours of optimization. After that point little or none improvement of the objective function can be spotted. This is also correlated to the size of the problem as stated above.

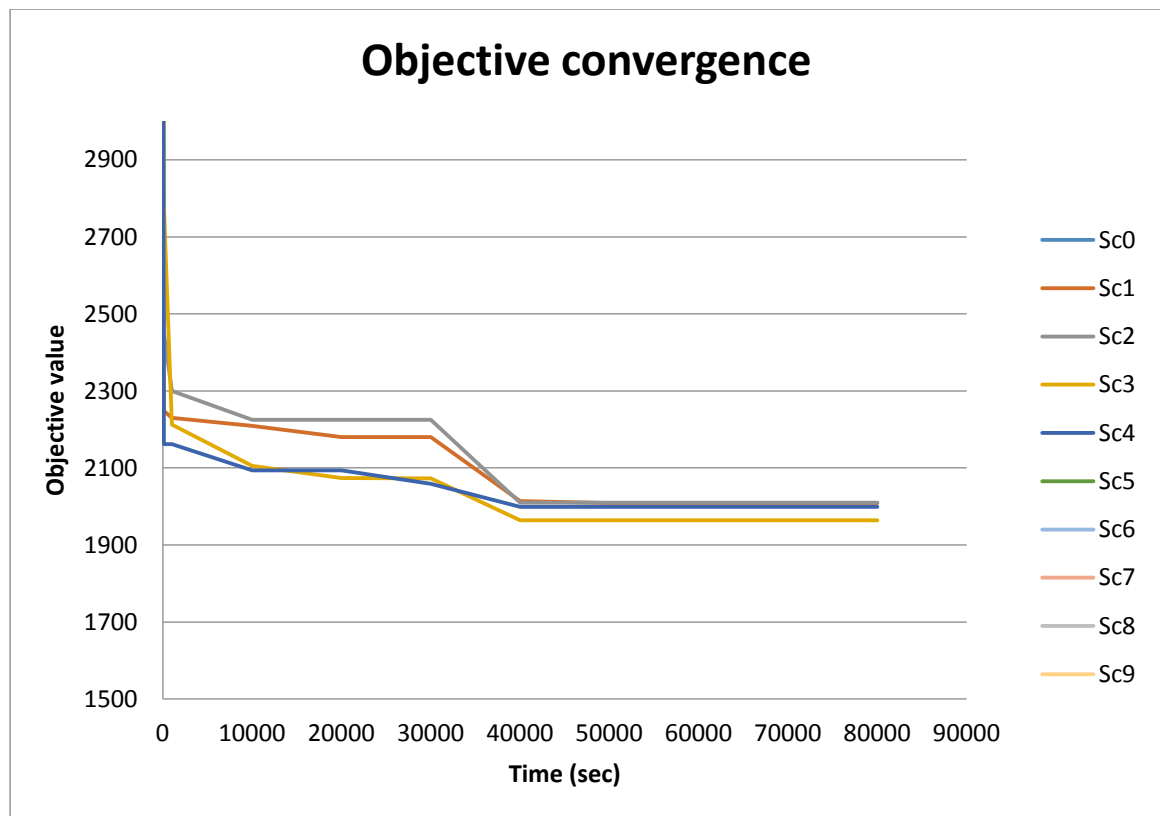


Figure 11: Objective convergence for various scenarios

5.6. Results after 24hours

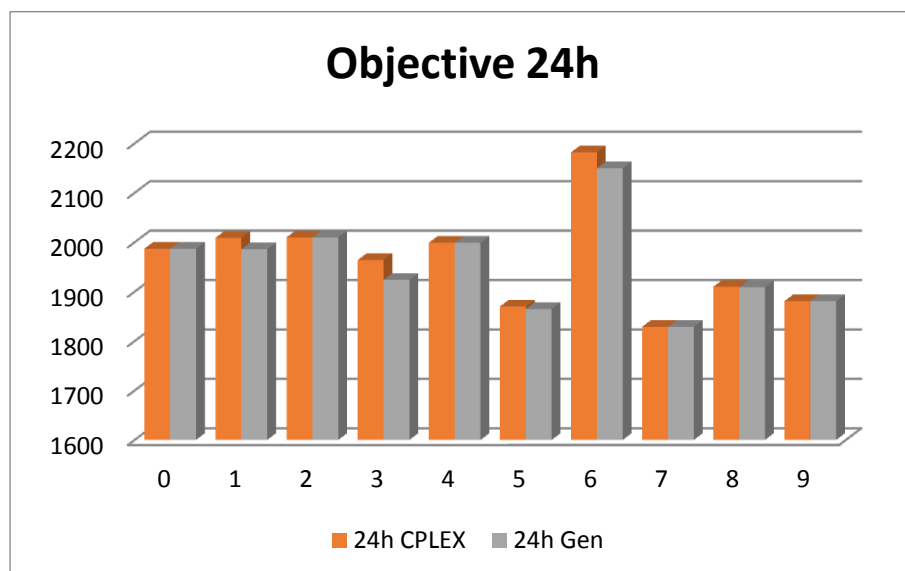


Figure 12: CPLEX and Genetic Algorithm results after 24h of optimization

As pointed above CPLEX optimization after some hours of optimizing is producing better feasible solutions with an significant decelerated pace where the size of the problem is radically enlarged. This behavior is expected to any NP-hard problem. New better feasible solutions are getting

harder to find as time passes by. Therefore there is a need to stop optimizing with CPLEX at some point and try to get better results with another method.

As mentioned above at the beginning CPLEX optimization was run for 24 hours, and then the solutions from the solution pool were mined and fed to the genetic algorithm as initial population for further optimization. As seen on Figure 12 the genetic algorithm managed at almost all different scenarios to produce a slightly better solution than the one produce by CPLEX. The optimization at scenario 1, 3 and 6 was significant improved using the genetic algorithm while at scenario 0, 4 and 8 failed to produce a better solution.

5.7. Results after 1,5 hours

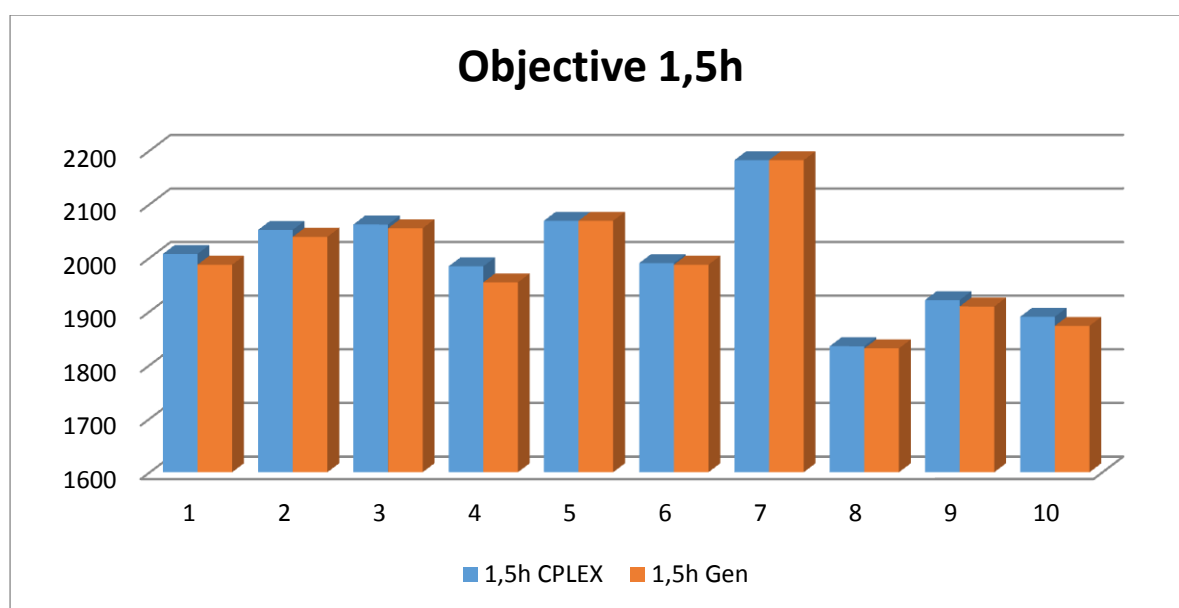


Figure 13: CPLEX and Genetic Algorithm results after 1.5h of optimization

The same procedure was repeated but CPLEX was allowed to run for only 1.5 hours. The solutions from the solution pool were again fed to the genetic algorithm as the initial population and the results as shown at Figure 13 suggest that almost at all scenarios the genetic algorithm managed to improve the optimization significantly. Regarding scenario 5, 6 and 7 CPLEX's solution pool produced only a few results and the initial population was proven not to be adequate for the genetic algorithm. Genetic Algorithm was run for 5 minutes approximately. The feasible solution discovered were in most cases known feasible solution that would be discovered from CPLEX after many hours of optimization. For example, scenarios 4, 9 and 10 genetically optimized to a better feasible solution than the one CPLEX discovered after 24h of running. The combination of 1.5h of CPLEX optimization and the genetic algorithm used, also managed to discover feasible solutions close to the ones produced after 24h of optimization for almost all other scenarios.

5.8. Results after 5 minutes

The above process was repeated one additional time. This time CPLEX optimization was executed for 5 minutes only. Again the solutions from the solution pool were extracted and fed to the genetic algorithm. The resulting optimization of the scenarios is presented at Figure 14

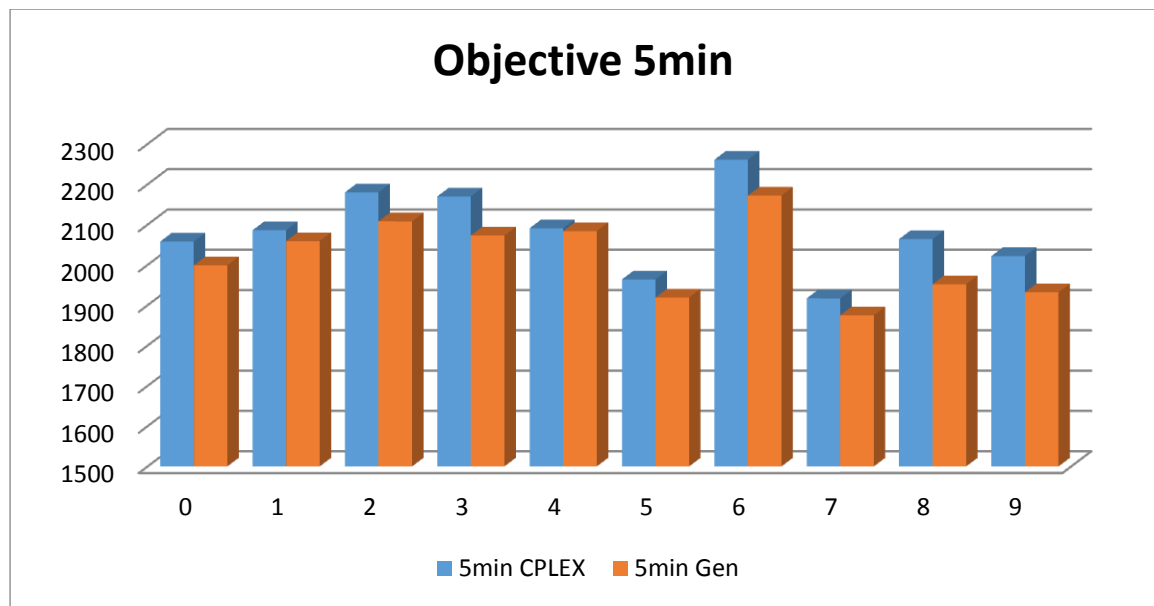


Figure 14: CPLEX and Genetic Algorithm results after 5min of optimization

As it can easily be spotted at each of the scenarios optimization from the genetic algorithm was also significant. The genetic algorithm managed to improve the initial population of solution providing good feasible solutions in short optimization time.

5.9. Comparison of obtained solutions

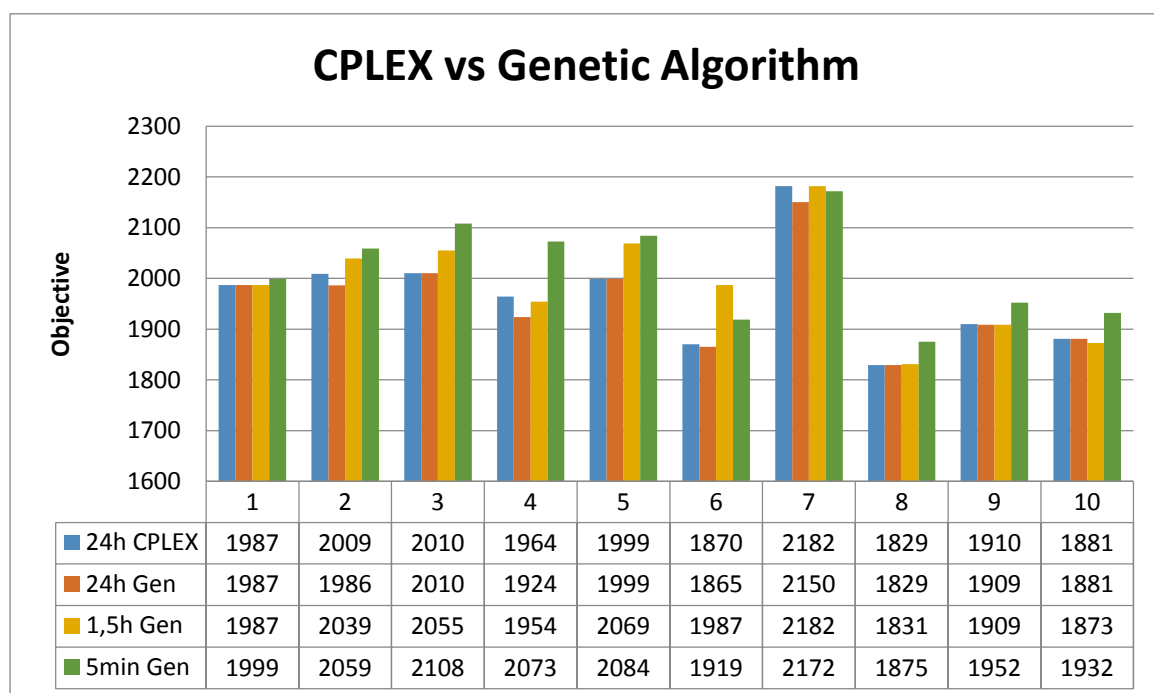


Figure 15: The results for each scenario

Lastly at Figure 15 CPLEX optimization output is compared against the optimization output at the genetic algorithm as executed with three different initial populations. The first bar indicates the best value obtained after 24 hours of CPLEX optimization while the next three are the mixed output of genetic optimization using as initial population the solutions from 24h, 1.5h and 5

minutes running of CPLEX. Even genetic optimization with an uneducated and primitive initial population returned rather promising results.

5.10. Conclusions

The mathematical formulation was run with CPLEX for various scenarios. The solution pool was then accessed and the feasible solutions were fed to the genetic algorithm as initial population. The results were then compared to the results of a 24h CPLEX optimization. The genetic algorithm although primitive in terms of crossover complexity proved to be useful, as it could provide in minutes feasible solutions that could be obtained after hours of CPLEX optimization.

6. Discussion

The vehicle routing problem is and will be for the next years a promising area for research. At this thesis a mathematical model was introduced for the multi depot vehicle routing problem. The solution of this problem will be presented and proposed to the Independent Power Transmission Operator of Greece (IPTO), in order to improve the substation inspection process. As stated in the thesis this inspection is a major part of the stability and readiness of an electric system.

The mathematical formulation of the problem, including the particularities of the routing as proposed by IPTO, were hard coded in C++ and run using CPLEX optimization software.

A genetic algorithm was also built in C++. The code was fed with solutions from CPLEX optimization solution pool, and the code was executed in order to improve the optimization.

It was noted that almost at all scenarios used to test the problem the hybrid genetic algorithm managed to improve the objective value. For instance the feasible solutions of 1,5 hours of optimization were fed as initial population at the genetic algorithm and produced a better feasible solution in a few minutes that in CPLEX would need 20 or more hours of optimization.

Improvements were achieved even when the feasible solution of the scenarios already optimized for 24 hours in CPLEX were fed in genetic algorithm.

The Genetic algorithm used in this thesis used a rather simple crossover and mutation function, but was executed with an initial population of feasible solutions that were mined from CPLEX optimization. This combination proved to be an extremely powerful tool.

Further better results can be achieved if the crossover and mutation functions are enhanced with techniques described in more recent research papers, especially for the MDVRP problem.

Lastly the genetic algorithm could also be enhanced with a more advance educating procedure that would optimize chromosomes in respect to the data of the problem.

7. Bibliography

- [1] J. Lenstra and A. Rinnooy Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, pp. 221-228, Vol 11 1981.
- [2] L. Bodin, A. Assad and M. Ball, "Routing and scheduling of vehicles and crews: The state of the art," *Computers and Operations Research*, Vol 10 1983.
- [3] C. Y. Liong, R. Wan, O. Khairuddin and M. Zirou, "Vehicle routing problem: Models and solutions," *Journal of Quality Measurement and Analysis*, pp. 205-218, 4 1 2008.
- [4] F. A. Tillman and T. M. Cain, "An upperbound algorithm for the single and multiple terminal delivery problem," *Management Science*, pp. 664-682, 18 11 1972.
- [5] A. Wren and A. Holliday, "Computer scheduling of vehicles from one or more depots to a number of delivery points," *Operational Research Quarterly*, pp. 333-344, 23 3 1972.
- [6] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, pp. 568-581, 12 4 1964.
- [7] W. Ho, G. S. Ho, P. Ji and H. C. Lau, "A hybrid genetic algorithm for the multi-depot vehicle routing problem," *Engineering Applications of Artificial Intelligence*, pp. 548-557, 21 4 2008.
- [8] B. Golden, S. Raghavan and E. Wasil, "Multi-terminal vehicle-dispatch algorithm," *Omega*, pp. 711-718, 4 1976.
- [9] O. M. Raft, "A modular algorithm for an extended vehicle scheduling problem," *European Journal of Operational Research*, pp. 67-76, 1 11 1982.
- [10] I. M. Chao, B. L. Golden and E. Wasil, "A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions.," *American Journal of Mathematical and Management Sciences*, pp. 371-406, 13 3 1993.
- [11] B. Golden, T. Magnanti and H. Nguyen, "Implementing vehicle routing algorithms," *Networks*, pp. 113-148, 7 1977.
- [12] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-torecord travel," *Journal of Computational Physics*, pp. 1086-1094, 1993.
- [13] J. Renaud, G. Laporte and F. F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem.," *Computers and Operational Research*, pp. 229-235, 1996.
- [14] J. F. Cordeau, M. Gendreau and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, pp. 105-119, 1997.

- [15] F. Glover, "Tabu search - part ii," *ORSA Journal on Computing*, pp. 4-32, 1 2 1990.
- [16] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems.," *Computers & Operations Research*, pp. 2403-2435, 34 8 2007.
- [17] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation Science*, pp. 455-472, 40 4 2006.
- [18] M. Mirabi, S. M. Fatemi Ghomi and F. Jolai, "Efficient stochastic hybrid heuristics for the multi-depot vehicle routing problem," *Robotics and Computer-Integrated Manufacturing*, p. 564–569, 26 6 2010.
- [19] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi and W. Rei, "A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems," CIRRELT, University of Montreal, 2011.
- [20] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers and Operations Research*, p. 1985–2002, 31 12 2004.
- [21] D. Martland, "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Genetic_algorithm. [Accessed 25 2 2015].
- [22] S. N. Sivanandam and S. N. Deepa, Introduction to genetic algorithms.
- [23] M. Obitko, "XI. Crossover and Mutation," [Online]. Available: <http://www.obitko.com/>. [Accessed 7 4 2011].
- [24] R. Dondo and J. Cerda, "A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows," *European Journal of Operational Research*, pp. 1478-1507, 176 2006.
- [25] Google, "Google Maps Service," [Online]. Available: <https://developers.google.com/maps/documentation/javascript/examples/distance-matrix>. [Accessed 10 10 2014].
- [26] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, p. 80–91, 1 6 1959.
- [27] M. Gendreau, J. Y. Potvin, O. Braysy, G. Hasle and A. Løkketangen, "Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography," in *The Vehicle Routing Problem: Latest Advances and Challenges*, 2008, pp. 143-169.
- [28] D. E. Gollberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley, 1989.
- [29] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, pp. 159-195, 2 9 2001.
- [30] J. H. Holland, Adaptation in natural and artificial systems, Ann Arbor: University of Michigan Press, 1975.

- [31] A. M. Law, *Simulation Modeling and Analysis*, McGraw-Hill, 2007.
- [32] J. Zhang, J. Tang and F. Y. Fung, "A scatter search for multi-depot vehicle routing problem with weight-related cost.," *Asia-Pacific Journal of Operations Research*, p. 323–348, 28 3 2011.

8. Appendix

8.1. The Google Matrix Code

Instance of code that gathers the distances from Google Maps, using the Google Maps Matrix Service.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Distance Matrix service</title>
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>
    <style>
      html, body {
        height: 100%;
        margin: 0;
        padding: 0;
      }

      #map-canvas {
        height: 100%;
        width: 50%;
      }
      #content-pane {
        float:right;
        width:48%;
        padding-left: 2%;
      }
      #outputDiv {
        font-size: 13px;
      }
      #beforeDiv {
        font-size: 11px;
      }
    </style>
  </script>
  var map;
  var geocoder;
  var bounds = new google.maps.LatLngBounds();
  var markersArray = [];
  var oname = new Array();
  var dname = new Array();
  var dest = new Array();
  var orig = new Array();

  var group ;

  oname.push("Farsala");
  orig.push(new google.maps.LatLng(39.33461276584468,22.366340160369873));

  group =2;


  if (group==1) {
    dname.push("Larisa1");
    dest.push(new google.maps.LatLng(39.62735832028149,22.39436388015747));

    dname.push("Volos2");
    dest.push(new google.maps.LatLng(39.369135530461925,22.966991364955902));

    dname.push("Lamia");
    dest.push(new google.maps.LatLng(38.92623064115468,22.420241832733154));

    dname.push("Trikala1");
    dest.push(new google.maps.LatLng(39.56311743957074,21.772810220718384));

    dname.push("Larisa2");
    dest.push(new google.maps.LatLng(39.66497155016516,22.44454264640808));
```

```
dname.push("Larisa3");
dest.push(new google.maps.LatLng(39.64791471390074,22.36564815044403));

dname.push("Larisa4");
dest.push(new google.maps.LatLng(39.58031508159118,22.448147535324097));

dname.push("Elassona");
dest.push(new google.maps.LatLng(39.85312600079744,22.168511152267456));

dname.push("Makrihori");
dest.push(new google.maps.LatLng(39.77323574729253,22.473435401916504));

dname.push("Platamonas");
dest.push(new google.maps.LatLng(40.02665322417621,22.563868761062622));

dname.push("Agia");
dest.push(new google.maps.LatLng(39.68771295466917,22.687625885009766));

dname.push("Stefanovikeio");
dest.push(new google.maps.LatLng(39.44436274143481,22.70199179649353));

dname.push("Volos3");
dest.push(new google.maps.LatLng(39.372393001151146,22.87083148956299));

dname.push("Volos1");
dest.push(new google.maps.LatLng(39.379832154261344,22.926149368286133));

dname.push("Lafkos");
dest.push(new google.maps.LatLng(39.17306622612494,23.251265287399292));

dname.push("Eyxinoupoli");
dest.push(new google.maps.LatLng(39.20153105608282,22.719104290008545));

dname.push("Almiros");
dest.push(new google.maps.LatLng(39.1581140401411,22.787210941314697));

dname.push("Achladi");
dest.push(new google.maps.LatLng(38.89654407300118,22.7985942363739));

dname.push("Stilida");
dest.push(new google.maps.LatLng(38.90690572840966,22.58165180683136));

dname.push("Sperxeiada");
dest.push(new google.maps.LatLng(38.91990366818331,22.125542163848877));

dname.push("Domokos");
dest.push(new google.maps.LatLng(39.11445378178841,22.31051802635193));

}

if (group==2){

dname.push("Leontari");
dest.push(new google.maps.LatLng(39.17692541194385,22.16115117073059));

dname.push("Sofades");
dest.push(new google.maps.LatLng(39.31799759940623,22.163221836090088));

dname.push("Karditsa");
dest.push(new google.maps.LatLng(39.3181636046047,21.910901069641113));

dname.push("YHSPlastira");
dest.push(new google.maps.LatLng(39.32933484908068,21.814534664154053));

dname.push("Trikala2");
dest.push(new google.maps.LatLng(39.52360750228261,21.821765899658203));

dname.push("Kalampaka");
dest.push(new google.maps.LatLng(39.7338416092686,21.578500270843506));

dname.push("Vounaina");
dest.push(new google.maps.LatLng(39.52299508882006,22.21806764602661));

dname.push("Farsala");
```



```

dest.push(new google.maps.LatLng(39.33461276584468,22.366340160369873));
}

function initialize() {
    var opts = {
        center: new google.maps.LatLng(39.62,22.39),
        zoom: 9
    };
    map = new google.maps.Map(document.getElementById('map-canvas'), opts);
    geocoder = new google.maps.Geocoder();

    var beforeDiv = document.getElementById('beforeDiv');

    beforeDiv.innerHTML = '';
    for (var ii=0;ii<oname.length;ii++){
        for (var jj=0;jj<dname.length;jj++){
            beforeDiv.innerHTML += oname[ii] + ' to ' + dname[jj] + '<br>';
        }
    }
}

function calculateDistances() {
    var service = new google.maps.DistanceMatrixService();
    service.getDistanceMatrix(
        {
            origins: orig,
            destinations: dest,
            travelMode: google.maps.TravelMode.DRIVING,
            unitSystem: google.maps.UnitSystem.METRIC,
            avoidHighways: false,
            avoidTolls: false
        }, callback);
}

function callback(response, status) {
    if (status != google.maps.DistanceMatrixStatus.OK) {
        alert('Error was: ' + status);
    } else {

        var origins = response.originAddresses;
        var destinations = response.destinationAddresses;
        var outputDiv = document.getElementById('outputDiv');
        outputDiv.innerHTML = '';

        deleteOverlays();

        for (var i = 0; i < origins.length; i++) {
            var results = response.rows[i].elements;
            addMarker(origins[i], false);
            outputDiv.innerHTML += '<tr>';
            for (var j = 0; j < results.length; j++) {
                addMarker(destinations[j], true);
                outputDiv.innerHTML += oname[i] + '---->' + dname[j] + '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;';
                +results[j].distance.text + '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>' + Math.round(results[j].duration.value/60)
                + '&nbsp;&nbsp;&min</b><br>';
                if ((j+1)/4==Math.floor((j+1)/4)) outputDiv.innerHTML += '<hr>';
                if(j==9) outputDiv.innerHTML += '<hr><hr><hr>';
            }
        }
    }
}

function addMarker(location, isDestination) {

    geocoder.geocode({'address': location}, function(results, status) {
        if (status == google.maps.GeocoderStatus.OK) {
            bounds.extend(results[0].geometry.location);
            map.fitBounds(bounds);

```

```

        var marker = new google.maps.Marker({
            map: map,
            position: results[0].geometry.location,
        });
        markersArray.push(marker);
    } else {
        alert('Geocode was not successful for the following reason: '
            + status);
    }
    });
}

function deleteOverlays() {
    for (var i = 0; i < markersArray.length; i++) {
        markersArray[i].setMap(null);
    }
    markersArray = [];
}

google.maps.event.addDomListener(window, 'load', initialize);

</script>
</head>
<body>
    <div id="content-pane">
        <div id="beforeDiv">
        </div>
        <div id="inputs">
            <p><button type="button" onclick="calculateDistances();">Calculate
                distances</button></p>

        </div>
        <div id="outputDiv"></div>
    </div>
    <div id="map-canvas"></div>
</body>
</html>

```

8.2. Code to refine the distance matrix produced from google maps

The refinement of the data produced from Google maps was essential since there were several irregularities. For instance the distance (in minutes or km) from substation k to another was different than from s to k. This was introduced due to the way Google maps calculate the distances (actually solving a routing problem). The actual point of a substation and the connected road could be assumed as one way, or two way with double uninterrupted line, were actually it is a small agriculture road. In the first case (assumed from Google maps) the route going or leaving the substation could not be the same were actually it is the same.

The code introduced above eliminates such irregularities choosing the smallest distance (in minutes), and adding a small

```

#include <stdio.h>
#include <string.h>

void main()
{
    FILE *fr,*fw,*fn;
    int i, j, k=0;
    int temp;
    int km[29][29];
    char name[32][30];

```

```

//OPEN FILES
fr=fopen("apostasiologio.txt", "r");
if (fr==NULL) {
    printf("Cannot open file apostasiologio.txt\nExiting.....\n");
    exit();
}
fn=fopen("substations.txt", "r");
if (fn==NULL) {
    printf("Cannot open file substations.txt\nExiting.....\n");
    exit();
}
fw=fopen("distances.txt", "w+");
if (fr==NULL) {
    printf("Cannot open file distances.txt\nExiting.....\n");
    exit();
}

// READ SUBSTATION NAMES save them to name[i]
for(i=0;i<32;i++){
    fgets(name[i],30,fn);
}

// READ KM save them to km[32][32]
for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        fscanf(fr,"%i",&temp);
        km[i][j] = temp;
    }
}

// PRINT EVERYTHING
for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        printf("%i ",km[i][j]);
    }
    printf("\n");
}
for(i=0;i<29;i++){
    printf("%s",name[i]);
}

for(i=0;i<29;i++){
    for(j=i;j<29;j++){
        if (km[i][j]>km[j][i]){
            km[i][j]=km[j][i];
            k++;
        }
        if (km[i][j]<km[j][i])
        {
            km[j][i]=km[i][j];
            k++;
        }
    }
}

// PRINT EVERYTHING AFTER THE ALTERATIONS
for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        printf("%i ",km[i][j]);
    }
    printf("\n");
}

printf("/n/n ALLAGES=%i",k);

for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        fprintf(fw,"%i\t",km[i][j]);
    }
    fprintf(fw,"\n");
}
}

```

8.3. The mathematical model as coded for CPLEX (in C++)

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

int s,k,d,t;
const int smax=25;//number of all stations including centrals
const int kmax=25;//j = k
const int dmax=4; // Depots
int tmax; // Max days for Routing. Will be defined according to the substations

const int BigM=10000;
const float SmallE=0.0001;

int main (int argc, char **argv)
{
//Πίνακες για την αποθήκευση των τιμών κόστους μετάβασης μεταξύ υποσταθμών και depot
int c[smax+dmax][smax+dmax],cl[smax+dmax][smax+dmax], assigned[smax][dmax];

int m[smax]; //πίνακας για την διάρκεια συντήρησης ανά υποσταθμό
int ml[smax]; //πίνακας για την διάρκεια συντήρησης ανά υποσταθμό
int sd[smax]; //Πίνακας για τους σταθμούς που δεν θα μπουν στο πρόγραμμα
int MT[dmax]; //Μέγιστος χρόνος route ανά depot
int i,j,il,jl,mindep, mindist; //διάφοροι δείκτες
int n, flag;//n = υποσταθμοί εντός προγράμματος επίσκεψης
string ff, ff1, ff2;
// ***** POPULATE
float pop_sd[smax];
float pop_st[smax];
int pop_final[smax];
//*****

// Ορισμός μέγιστου χρόνου route ανά depot
MT[0]=360;
MT[1]=360;
MT[2]=360;
MT[3]=360;

// όνομα του αρχείου αποτελεσμάτων
ff="36H - No heuristic Tricks - All Substations - Solution Pool";
ff1=ff+".txt";

n=0;

ofstream result;
ofstream genes;
genes.open("genes-36h-all-noheuristic.txt",ios::out);
if (genes.bad())
cout <<endl<<endl<<" CANNOT OPEN genes4.txt "<<endl<<endl;

result.open(ff1.c_str(), ios::out);

if (result.bad())
cout << "\n\n Cannot open output file";

//-----READ SDs matrix for not routing to specified substations-----
ifstream datafile3;
datafile3.open ("sds.txt", ios::in);
if (datafile3.bad()) cout << "\n----\nCannot open sds.txt\n-----\n";
else
{
for ( i = 0; i < smax; i++) // διάβασε για όλους τους υποσταθμούς
{
```

```

        datafile3 >> sd[i];
        if (sd[i]==0) n++; // μέτρα στο n αυτούς που είναι προς επίσκεψη
    }
    datafile3.close();
}
result<<ff<<"\n\n";
ff2=ff+"-dist.txt"; //ονομα αρχείου αποστάσεων

result << " Substations: " << n << "\n";

tmax=10;

//-----READ DISTANCES FROM FILE distances.txt-----The first dmax lines are depots-----
ifstream datafile;
ofstream datafile1;
datafile.open ("distances-r.txt", ios::in);
datafile1.open (ff2.c_str(), ios::out);
if (datafile.bad())
    cout << "\n----\nCannot open distances.txt\n-----\n";
else
{
    for ( i = 0; i < smax+dmax; i++)
    {
        for ( j = 0; j < smax+dmax; j++)
        {
            datafile >> c1[i][j]; // αποθήκευσε τις αποστάσεις στον c1
        }
    }
    datafile.close();

    // Βάζω δεδομένα στο c μόνο για τους υποσταθμούς που είναι προς συντήρηση
    flag=0;
    i1=0; // δείκτης γραμμής για τον νέο πίνακα
    for (i=0;i<smax+dmax;i++)
    {
        j1=0; // δείκτης στήλης για τον νέο πίνακα
        flag=0;
        for(j=0;j<kmax+dmax;j++)
        {
            // περιπτώσεις που το c1 στοιχείο θα αποθηκευτεί στον c πίνακα
            if((i<dmax && j<dmax) || (i>=dmax && j<dmax && sd[i-dmax]==0) || (j>=dmax && i<dmax &&
sd[j-dmax]==0) || (i>=dmax && j>=dmax && sd[i-dmax]==0 & sd[j-dmax]==0))
            {
                c[i1][j1++]=c1[i][j];
                flag=1;
            }
        }
        if (flag==1) // αλλαγή γραμμής νέου πίνακα
        {
            i1++;
        }
    }
}

/*
for (i=dmax;i<n+dmax;i++)
{
    mindist=5000;
    for (j=0;j<dmax;j++)
    {
        if (c[i][j]<mindist)
        {
            mindist=c[i][j];
            mindep=j;
        }
    }
    for (j1=i;j1<n+dmax;j1++)
    {
        if (2*c[i][j1]<c[i][mindep] && i!=j1)
        {

```

```

        cout << "\n" << "Possible : "<< i << " and " << j1 << " ->" << c[i][j1] << " , " <<
c[i][mindep] ;
    }
}
*/

il=0;
//-----READ cost visiting s substation ----- The first dmax lines are depots-----
ifstream datafile2;

datafile2.open ("ms.txt", ios::in);
if (datafile2.bad())
    cout << "\n----\nCannot open ms.txt\n-----\n";
else
{
    for ( i = 0; i < smax; i++){
        datafile2 >> m1[i];
        if (sd[i]==0){ // μόνο τα κόστη των σταθμών που είναι προς επίσκεψη
            m[i]=m1[i];
            result<<i<<"\t"<<m[i]<<"\n";
            il++;
        }
    }
    datafile2.close();
}

// Δημιουργία του -dist.txt για την επαλήθευση των στοιχείων
for (i=0;i<n+dmax;i++)
{
    if (i==0) // πρώτη γραμμή
    {
        datafile1 << "\t";
        for (il=0;il<dmax;il++) datafile1 << "d" << il << "\t"; // γράψε τα d σαν τίτλοι
        for (il=0;il<n;il++) datafile1 << "s" << il << "\t"; // γράψε τα s σαν τίτλοι
        datafile1 << "\t" << "m"; //γράψε m σαν τίτλο
        datafile1 << "\n";
    }
    for(j=0;j<n+dmax;j++)
    {
        // σε κάθε γραμμή προηγείται είτε το d
        if (i<dmax && j==0) datafile1 << "d" << i << "\t";
        // είτε το s ανάλογα την γραμμή
        if (i>=dmax && j==0) datafile1 << "s" << i-dmax << "\t";
        datafile1 << c[i][j] << "\t"; //ακολουθούν τα στοιχεία κόστους της γραμμής
        if (j==n+dmax-1 && i>=dmax) datafile1 << "\t"<<m[i-dmax] ;
    }
    datafile1 << "\n"; //αλλαγή γραμμής
}
datafile1.close();

//cout<<"Swag ..."<<"\n";
//-----
IloEnv env;

try {

IloModel model (env);

typedef IloArray<IloNumArray> IloNumMatrix2x2;
typedef IloArray<IloNumMatrix2x2> IloNumMatrix3x3;
typedef IloArray<IloNumMatrix3x3> IloNumMatrix4x4;
typedef IloArray<IloNumMatrix4x4> IloNumMatrix5x5;

typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
typedef IloArray<IloNumVarMatrix2x2> IloNumVarMatrix3x3;
typedef IloArray<IloNumVarMatrix3x3> IloNumVarMatrix4x4;
typedef IloArray<IloNumVarMatrix4x4> IloNumVarMatrix5x5;

```

```

typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
typedef IloArray<IloRangeMatrix2x2> IloRangeMatrix3x3;
typedef IloArray<IloRangeMatrix3x3> IloRangeMatrix4x4;
typedef IloArray<IloRangeMatrix4x4> IloRangeMatrix5x5;

IloCplex cplex(env);

//----- Decision Variable CV -----

IloNumVarArray CVs(env,0);
for (s=0;s<n;s++){
    char COSTUPTO_S[70];
    sprintf(COSTUPTO_S,"CVs(s%d)",s);
    IloNumVar CV(env,0,5000,ILOFLOAT,COSTUPTO_S);
    CVs.add(CV);
}

//----- Decision Variable X -----

IloNumVarMatrix2x2 Xst(env,0);
for (s=0;s<n;s++){
    IloNumVarArray Xt(env,0);
    for (t=0;t<tmax;t++){
        char Binary_X[70];
        sprintf(Binary_X,"Xst(s%d,t%d)",s,t);
        IloNumVar X(env,0,1,ILOINT,Binary_X);
        Xt.add(X);
    }
    Xst.add(Xt);
}

//----- Decision Variable Y -----

IloNumVarMatrix2x2 Ysd(env,0);
for (s=0;s<n;s++){
    IloNumVarArray Yd(env,0);
    for (d=0;d<dmax;d++){
        char Binary_Y[70];
        sprintf(Binary_Y,"Ysd(s%d,d%d)",s,d);

        if (s<=40){ // for float relaxation
            IloNumVar Y(env,0,1,ILOINT,Binary_Y);
            Yd.add(Y);
        } else
        {
            IloNumVar Y(env,0,1,ILOFLOAT,Binary_Y);
            Yd.add(Y);
        }
    }
    Ysd.add(Yd);
}

//----- Decision Variable R-----

IloNumVarMatrix2x2 Rsk(env,0);
for (s=0;s<n;s++){
    IloNumVarArray Rk(env,0);
    for (k=0;k<n;k++){
        char Binary_R[70];
        sprintf(Binary_R,"Rsk(s%d,k%d)",s,k);
        IloNumVar R(env,0,1,ILOINT,Binary_R);
        Rk.add(R);
    }
    Rsk.add(Rk);
}

```

```

//----- Decision Variable CVT -----

IloNumVarMatrix2x2 CVTdt(env,0);
for(d=0;d<dmax;d++){
IloNumVarArray CVTt(env,0);
for (t=0;t<tmax;t++){
char Totalcostfor_tday[70];
sprintf(Totalcostfor_tday,"CVTdt(d%d,t%d)",d,t);
IloNumVar CVT(env,0,5000,ILOFLOAT,Totalcostfor_tday);
CVTt.add(CVT);
}
CVTdt.add(CVTt);
}

//-----
//-----CONSTRAINTS-----

// -----
// Each substation is assigned to 1 depot
//-----

IloRangeArray Ys(env,0);
for(s=0;s<n;s++){
IloExpr expr(env,0);
for(d=0;d<dmax;d++){
expr += Ysd[s][d];
}
char Binary_Ys[60];
sprintf(Binary_Ys,"Ys(s%d,d%d)",s,d);
float LB=1,UB=1;
IloRange Y(env,LB,expr,UB,Binary_Ys);
expr.end();
model.add(Y);
Ys.add(Y);
}

// -----
// Each substation is visited only once
//-----

IloRangeArray Xs(env,0);
for(s=0;s<n;s++){
IloExpr expr(env,0);
for(t=0;t<tmax;t++){
expr += Xst[s][t];
}
char Binary_Xs[60];
sprintf(Binary_Xs,"Xs(s%d,t%d)",s,t);
float LB=1,UB=1;
IloRange X(env,LB,expr,UB,Binary_Xs);
expr.end();
model.add(X);
Xs.add(X);
}

//-----
//----- Cost until 1st substation -----
//-----

IloRangeMatrix3x3 CVsdt(env,0);
for(s=0;s<n;s++){
IloRangeMatrix2x2 CVdt(env,0);
for(d=0;d<dmax;d++){
IloRangeArray CVt(env,0);
for (t=0;t<tmax;t++){
IloExpr expr(env,0);
expr += CVs[s]-(c[d][s+dmax]+m[s])*(Xst[s][t]+Ysd[s][d]-1);
char CostToS[60];

```



```

sprintf(CostToS, "CVs(s%d,d%d,t%d)", s, d, t);
float LB=0, UB=IloInfinity;
IloRange CV(env, LB, expr, UB, CostToS);
expr.end();
model.add(CV);
CVt.add(CV);
}
CVdt.add(CVt);
}
CVsdt.add(CVdt);
}

//-----
//----- Cost until k substation -----
//-----

IloRangeMatrix3x3 CV2kst(env, 0);
for(s=0; s<n; s++){
    IloRangeMatrix2x2 CV2st(env, 0);
    for(k=0; k<s; k++){
        IloRangeArray CV2t(env, 0);
        for (t=0; t<tmax; t++){

            IloExpr expr(env, 0);
            expr += CVs[k]-CVs[s]-(c[s+dmax][k+dmax]+m[k])+BigM*(1-Rsk[s][k])+BigM*(2-Xst[s][t]-Xst[k][t]);

            char CostToK[60];
            sprintf(CostToK, "CV2k(k%d,s%d,t%d)", k, s, t);
            float LB=0, UB=IloInfinity;
            IloRange CV2(env, LB, expr, UB, CostToK);
            expr.end();
            model.add(CV2);
            CV2t.add(CV2);
        }
        CV2st.add(CV2t);
    }
    CV2kst.add(CV2st);
}

//-----
//----- Cost until s from k substation -----
//-----

IloRangeMatrix3x3 CV3kst(env, 0);
for(s=0; s<n; s++){
    IloRangeMatrix2x2 CV3st(env, 0);
    for(k=0; k<s; k++){

        IloRangeArray CV3t(env, 0);
        for (t=0; t<tmax; t++){

            IloExpr expr(env, 0);
            expr += CVs[s]-CVs[k]-(c[s+dmax][k+dmax]+m[s])+BigM*Rsk[s][k]+BigM*(2-Xst[s][t]-Xst[k][t]);

            char CostToS[60];
            sprintf(CostToS, "CV3k(k%d,s%d,t%d)", k, s, t);
            float LB=0, UB=IloInfinity;
            IloRange CV3(env, LB, expr, UB, CostToS);
            expr.end();
            model.add(CV3);
            CV3t.add(CV3);
        }
        CV3st.add(CV3t);
    }
    CV3kst.add(CV3st);
}

//-----
//----- Cost until d-depot -----
//-----

```

```

IloRangeMatrix3x3 CVT1dts (env,0);
for(d=0;d<dmax;d++){
  IloRangeMatrix2x2 CVT1ts (env,0);
  for(t=0;t<tmax;t++){
    IloRangeArray CVT1s (env,0);
    for (s=0;s<n;s++){
      IloExpr expr (env,0);
      expr += CVTdt[d][t]-CVs[s]-c[s+dmax][d]+BigM*(2-Xst[s][t]-Ysd[s][d]);
      char CostToD[60];
      sprintf(CostToD,"CVT1dts (d%d,t%d,s%d)",d,t,s);
      float LB=0,UB=IloInfinity;
      IloRange CVT1 (env,LB,expr,UB,CostToD);
      expr.end();
      model.add(CVT1);
      CVT1s.add(CVT1);
    }
    CVT1ts.add(CVT1s);
  }
  CVT1dts.add(CVT1ts);
}

//-----
//----- Total Cost -----
//-----

IloRangeMatrix2x2 CVT3dt (env,0);
for(d=0;d<dmax;d++){
  IloRangeArray CVT3t (env,0);
  for (t=0;t<tmax;t++){
    IloExpr expr (env,0);
    expr += CVTdt[d][t]-MT[d];
    char Cost[60];
    sprintf(Cost,"CVT3dt (d%d,t%d)",d,t);
    float LB=-IloInfinity,UB=0;
    IloRange CVT3 (env,LB,expr,UB,Cost);
    expr.end();
    model.add(CVT3);
    CVT3t.add(CVT3);
  }
  CVT3dt.add(CVT3t);
}

//-----
//-----Objective Function -----
//-----
IloExpr expr (env,0);

for(d=0;d<dmax;d++){
  for(t=0;t<tmax;t++){
    expr+= CVTdt[d][t];
  }
}

model.add(IloMinimize (env, expr));
expr.end();

cplex.extract(model);
cplex.exportModel("otinanai.lp");

cplex.setParam(IloCplex::EpGap, 0.6); //stop when you succeed a % GAP
cplex.setParam(IloCplex::NodeFileInd, 3); // compress and use disk cache memory
cplex.setParam(IloCplex::WorkDir, "c:/cplextemp"); // directory of the cache memory

```

```

//cplex.setParam(IloCplex::Threads, 5); //number of threads to use
cplex.setParam(IloCplex::TiLim, 36*60*60); // time limit to end optimization regardless of GAP
or solutions
cplex.setOut(result); //write the results to a file
//Heuristic at branching
//cplex.setParam(IloCplex::LBHeur, 0);
//cplex.setParam(IloCplex::HeurFreq, 50000);

//cplex.setParam(IloCplex::RINSHeur,6);
//cplex.setParam(IloCplex::SubMIPNodeLim, 33);

//solution pool parameters
cplex.setParam(IloCplex::PopulateLim, 40000);
cplex.setParam(IloCplex::SolnPoolIntensity, 1);
cplex.setParam(IloCplex::SolnPoolCapacity, 1000000);
cplex.setParam(IloCplex::SolnPoolReplace, 1);
cplex.setParam(IloCplex::SolnPoolAGap, 2200);

cplex.solve();

cplex.populate();

env.out() <<endl<<endl<< "-----SOLUTION POOL RELATED -----"<<endl<<endl<<endl;

int numsol=cplex.getSolnPoolNsolns();
env.out() << "The solution pool contains " << numsol << " solutions." << endl;


float g,g1;
int pop_sd_n=0, pop_st_n=0, pop_final_n=0, pop_i, pop_j;
int sort_temp, sort_h;
for (int i = 0; i < numsol; i++)
{
    if (cplex.getObjValue(i)<4000)
    {
        result<<endl<<endl;
        result<< "Solution " << i << " with objective " << cplex.getObjValue(i);
        result<<"-----"<<endl;
        for(d=0;d<dmax;d++)
        {
            for(t=0;t<tmax;t++)
            {
                g = cplex.getValue(CVTdt[d][t],i);
                if(g>0.001)
                {
                    result<<"CVTdt"<<" ("<<d<<" "<<t<<" "<<"="<<g<<endl;
                    //*****
                    pop_sd_n=0;
                    pop_st_n=0;
                    pop_final_n=0;
                    for (s=0;s<n;s++)
                    {
                        g=cplex.getValue(Ysd[s][d],i);
                        if(g>0.001) pop_sd[pop_sd_n++]=s;
                    }

                    for (s=0;s<n;s++)
                    {
                        g=cplex.getValue(Xst[s][t],i);
                        if(g>0.001) pop_st[pop_st_n++]=s;
                    }

                    for (pop_i=0;pop_i<pop_sd_n;pop_i++)
                        for (pop_j=0;pop_j<pop_st_n;pop_j++)

```

```

        if (pop_sd[pop_i]==pop_st[pop_j]) pop_final[
pop_final_n++]=pop_sd[pop_i];
        result<<"substations: **before sorting ** : ";
        for (pop_i=0;pop_i<pop_final_n;pop_i++)
            result<<pop_final[pop_i]<<" ";
        result<<endl;

        // Sorting
        for (pop_i=0;pop_i<pop_final_n;pop_i++)
        {
            g=cplex.getValue(CVs[pop_final[pop_i]],i);
            sort_temp=pop_i;
            for (pop_j=pop_i;pop_j<pop_final_n;pop_j++)
            {
                g1=cplex.getValue(CVs[pop_final[pop_j]],i);
                if (g1<g)
                {
                    sort_temp=pop_j;
                    g=g1;
                }
            }
            if (sort_temp!=pop_i)
            {
                sort_h=pop_final[pop_i];
                pop_final[pop_i]=pop_final[sort_temp];
                pop_final[sort_temp]=sort_h;
            }
        }
        // End of Sorting

        genes<<(d+100)<<" ";
        result<<"substations: **after sorting ** : ";
        for (pop_i=0;pop_i<pop_final_n;pop_i++)
        {
            result<<pop_final[pop_i]<<" ";
            genes<<pop_final[pop_i]<<" ";
        }
        result<<endl;
        genes<<(d+100)<<" ";
        //*****
    }
}

result<<endl;
for(s=0;s<n;s++)
{
    for(t=0;t<tmax;t++)
    {
        g = cplex.getValue(Xst[s][t],i);
        if(g>0.001) result<<"Xst"<<" ("<<s<<" "<<t<<" "<<"="<<g<<endl;
    }
}
result<<endl;
for(s=0;s<n;s++)
{
    for(d=0;d<dmax;d++)
    {
        g = cplex.getValue(Ysd[s][d]);
        if(g>0.001) result<<"Ysd"<<" ("<<s<<" "<<d<<" "<<"="<<g<<endl;
    }
}
result<<endl;
for(s=0;s<n;s++)
{
    g = cplex.getValue(CVs[s],i);
    result<<"CVs"<<" ("<<s<<" "<<"="<<g<<endl;
}
genes<<endl;
}

}

if (!cplex.solve ()) {
env.error()<<"Failed to optimize LP."<<endl;
throw(-1);
}

```

```

}

result<<"Solution status = " <<cplex.getStatus()<<endl;
result<<"Solution value = " <<cplex.getObjValue()<<endl;

//-----ektypwsh metavlhtwn-----CVTdt
result<<"\n-----CVTdt-----"<<endl;
for (d=0;d<dmax;d++){
for (t=0;t<tmax;t++){
g = cplex.getValue(CVTdt[d][t]);
if (g!=0) result<<"CVTdt"<<"("<<d<<","<<t<<")"<<"="<<g<<endl;
}
}

//-----ektypwsh metavlhtwn-----Xst
result<<"\n-----Xst-----"<<endl;
for (s=0;s<n;s++){
for (t=0;t<tmax;t++){
g = cplex.getValue(Xst[s][t]);
if (g!=0) result<<"Xst"<<"("<<s<<","<<t<<")"<<"="<<g<<endl;
}
}

//-----ektypwsh metavlhtwn-----Ydt
result<<"\n-----Ysd-----"<<endl;
for (s=0;s<n;s++){
for (d=0;d<dmax;d++){
g = cplex.getValue(Ysd[s][d]);
if (g!=0) result<<"Ysd"<<"("<<s<<","<<d<<")"<<"="<<g<<endl;
}
}

//-----ektypwsh metavlhtwn-----Rsk
result<<"\n-----Rsk-----"<<endl;

for (s=0;s<n;s++){
for (k=0;k<s;k++){

g = cplex.getValue(Rsk[s][k]);
if (g!=0) result<<"Rsk"<<"("<<s<<","<<k<<")"<<"="<<g<<endl;
}
}

//-----ektypwsh metavlhtwn-----Cvs
result<<"\n-----CVs-----"<<endl;
for (s=0;s<n;s++)
{

g = cplex.getValue(CVs[s]);
result<<"CVs"<<"("<<s<<")"<<"="<<g<<endl;

}

//-----

}
catch ( IloException& e){
cerr << "concert exception caught:"<<e<<endl;
}
catch (...){
cerr<<"Unknown exception caught" <<endl;
}
env.end();

return 0;
} //End main

```

8.4. The distance file used:

Initial distances as calculated by Google Maps Matrix Process

		Larisa1	Volos2	Lamia	Trikala1	Larisa2	Larisa3	Larisa4	Elassona	Makrioni	Platamonas	Agia	Stefanovikeio	Volos3	Volos1	Lafkos	Eyxinoupoli	Almiros	Achladi	Stilida	Sperxeiada	Domokos	Leontari	Sofades	Karditsa	YHSPlastira	Trikala2	Kalampaka	Vounaina	Farsala
d0	Larisa1	0	50	87	49	13	21	10	50	30	54	42	29	44	46	109	56	53	71	87	121	65	72	67	71	76	51	70	33	39
d1	Volos2	50	0	88	96	49	63	47	92	59	81	64	26	14	8	65	43	41	59	75	116	91	98	94	115	120	97	117	78	69
d2	Lamia	89	89	0	94	92	106	85	135	106	128	115	76	81	83	142	66	61	33	18	34	24	50	56	78	83	85	114	99	57
d3	Trikala1	50	98	96	0	58	66	54	77	74	99	87	73	88	90	153	99	97	115	111	125	73	72	59	41	40	11	24	54	70
s0	Larisa2	24	55	97	68	0	29	24	58	17	42	30	30	45	47	111	57	55	73	88	130	74	81	78	90	95	70	89	52	48
s1	Larisa3	19	67	102	63	19	0	25	51	35	60	48	43	57	59	123	69	67	85	101	136	80	87	82	86	91	65	84	48	54
s2	Larisa4	16	51	80	62	18	32	0	61	31	53	39	26	40	42	106	52	50	68	84	114	57	64	60	77	82	64	83	43	31
s3	Elassona	47	96	131	77	48	49	54	0	54	89	77	71	86	88	152	98	95	114	129	165	108	116	109	107	110	78	98	65	83
s4	Makrioni	28	63	105	73	17	34	28	54	0	39	43	39	53	55	119	65	63	81	97	138	82	90	86	94	99	74	94	56	56
s5	Platamonas	56	89	130	101	45	61	56	89	43	0	68	64	79	81	144	90	88	106	122	163	108	115	111	122	127	102	121	84	82
s6	Agia	44	71	118	89	33	50	44	78	41	63	0	45	61	62	126	72	70	88	104	145	95	102	98	110	115	90	110	72	69
s7	Stefanovikeio	29	32	78	76	28	43	27	72	39	61	44	0	22	24	87	33	31	49	65	106	76	83	79	95	100	77	97	57	50
s8	Volos3	45	17	83	91	44	58	42	87	54	76	59	21	0	9	73	38	36	54	70	111	86	93	89	111	115	93	112	73	64
s9	Volos1	47	11	85	93	46	60	44	89	56	78	61	23	11	0	69	40	38	56	72	113	88	95	91	112	117	95	114	75	66
s10	Lafkos	110	62	146	157	110	124	108	153	120	142	125	87	75	69	0	102	99	117	133	174	149	156	152	174	179	158	178	138	127
s11	Eyxinoupoli	58	47	66	104	57	71	55	100	67	89	72	34	39	41	101	0	13	37	53	94	81	88	84	106	111	106	125	86	59
s12	Almiros	55	45	61	102	54	69	53	98	65	87	70	32	36	38	98	14	0	31	47	89	78	85	81	103	108	103	123	83	57
s13	Achladi	74	63	32	120	73	87	71	116	83	105	88	50	54	56	116	40	34	0	19	61	52	78	84	106	111	113	141	101	75
s14	Stilida	94	83	19	108	93	107	91	136	103	125	108	70	75	77	136	60	55	27	0	48	39	65	71	92	97	100	129	114	72
s15	Sperxeiada	124	118	35	123	126	140	120	169	138	160	143	105	109	111	171	95	89	61	46	0	59	69	90	92	106	115	144	129	92
s16	Domokos	66	94	25	70	68	82	61	111	83	105	91	75	87	89	148	79	77	55	40	59	0	27	32	54	59	62	91	76	34
s17	Leontari	73	102	51	69	76	90	69	119	90	113	99	83	95	97	155	87	84	81	66	68	28	0	32	38	52	61	90	74	41
s18	Sofades	67	105	61	55	75	84	71	111	91	116	104	86	98	100	158	90	87	91	76	90	38	36	0	39	44	47	76	60	41
s19	Karditsa	67	114	73	36	75	83	71	104	91	115	103	89	104	106	170	102	100	103	88	88	51	37	36	0	19	28	57	60	54
s20	YHSPlastira	75	122	82	41	83	92	79	109	99	124	111	97	112	114	178	110	108	111	96	104	59	54	45	20	0	33	62	68	63
s21	Trikala2	51	99	89	11	59	67	55	78	75	100	88	74	89	91	154	100	98	116	104	118	66	65	52	34	33	0	32	55	70
s22	Kalampaka	80	128	126	32	88	96	85	89	105	129	117	103	118	120	184	130	127	146	141	155	103	102	89	71	70	41	0	84	101
s23	Vounaina	34	81	101	55	42	51	38	66	58	83	71	57	71	73	137	83	81	99	114	130	78	77	60	64	69	56	75	0	52
s23	Farsala	39	74	55	71	41	55	35	84	56	78	64	49	63	65	127	59	56	75	70	89	32	39	35	57	62	65	92	52	0

The above numbers after fixing the irregularities were fed to the CPLEX optimization as follows:

0	50	87	49	13	19	10	47	28	54	42	29	44	46	109	56	53	71	87	121	65	72	67	67	75	51	70	33	39
50	0	88	96	49	63	47	92	59	81	64	26	14	8	62	43	41	59	75	116	91	98	94	114	120	97	117	78	69
87	88	0	94	92	102	80	131	105	128	115	76	81	83	142	66	61	32	18	34	24	50	56	73	82	85	114	99	55
49	96	94	0	58	63	54	77	73	99	87	73	88	90	153	99	97	115	108	123	70	69	55	36	40	11	24	54	70
13	49	92	58	0	19	18	48	17	42	30	28	44	46	110	57	54	73	88	126	68	76	75	75	83	59	88	42	41
19	63	102	63	19	0	25	49	34	60	48	43	57	59	123	69	67	85	101	136	80	87	82	83	91	65	84	48	54
10	47	80	54	18	25	0	54	28	53	39	26	40	42	106	52	50	68	84	114	57	64	60	71	79	55	83	38	31
47	92	131	77	48	49	54	0	54	89	77	71	86	88	152	98	95	114	129	165	108	116	109	104	109	78	89	65	83
28	59	105	73	17	34	28	54	0	39	41	39	53	55	119	65	63	81	97	138	82	90	86	91	99	74	94	56	56
54	81	128	99	42	60	53	89	39	0	63	61	76	78	142	89	87	105	122	160	105	113	111	115	124	100	121	83	78
42	64	115	87	30	48	39	77	41	63	0	44	59	61	125	72	70	88	104	143	91	99	98	103	111	88	110	71	64
29	26	76	73	28	43	26	71	39	61	44	0	21	23	87	33	31	49	65	105	75	83	79	89	97	74	97	57	49
44	14	81	88	44	57	40	86	53	76	59	21	0	9	73	38	36	54	70	109	86	93	89	104	112	89	112	71	63
46	8	83	90	46	59	42	88	55	78	61	23	9	0	69	40	38	56	72	111	88	95	91	106	114	91	114	73	65
109	62	142	153	110	123	106	152	119	142	125	87	73	69	0	101	98	116	133	171	148	155	152	170	178	154	178	137	127
56	43	66	99	57	69	52	98	65	89	72	33	38	40	101	0	13	37	53	94	79	87	84	102	110	100	125	83	59
53	41	61	97	54	67	50	95	63	87	70	31	36	38	98	13	0	31	47	89	77	84	81	100	108	98	123	81	56
71	59	32	115	73	85	68	114	81	105	88	49	54	56	116	37	31	0	19	61	52	78	84	103	111	113	141	99	75
87	75	18	108	88	101	84	129	97	122	104	65	70	72	133	53	47	19	0	46	39	65	71	88	96	100	129	114	70
121	116	34	123	126	136	114	165	138	160	143	105	109	111	171	94	89	61	46	0	59	68	90	88	104	115	144	129	89
65	91	24	70	68	80	57	108	82	105	91	75	86	88	148	79	77	52	39	59	0	27	32	51	59	62	91	76	32
72	98	50	69	76	87	64	116	90	113	99	83	93	95	155	87	84	78	65	68	27	0	32	37	52	61	90	74	39
67	94	56	55	75	82	60	109	86	111	98	79	89	91	152	84	81	84	71	90	32	32	0	36	44	47	76	60	35

67	114	73	36	75	83	71	104	91	115	103	89	104	106	170	102	100	103	88	88	51	37	36	0	19	28	57	60	54
75	120	82	40	83	91	79	109	99	124	111	97	112	114	178	110	108	111	96	104	59	52	44	19	0	33	62	68	62
51	97	85	11	59	65	55	78	74	100	88	74	89	91	154	100	98	113	100	115	62	61	47	28	33	0	32	55	65
70	117	114	24	88	84	83	89	94	121	110	97	112	114	178	125	123	141	129	144	91	90	76	57	62	32	0	75	92
33	78	99	54	42	48	38	65	56	83	71	57	71	73	137	83	81	99	114	129	76	74	60	60	68	55	75	0	52
39	69	55	70	41	54	31	83	56	78	64	49	63	65	127	59	56	75	70	89	32	39	35	54	62	65	92	52	0

8.5. The chromosomes exported from CPLEX optimization

Below there is the representation of the genes as calculated and been saved from the CPLEX code. Each chromosome is a feasible solution. The large numbers (100, 101, 102, 103) are the 4 depots. The small numbers are the substations visited. From instance 100 5 6 100 is a route commencing from depot 0, visiting substation 5, then substation 6 and returning to depot 0.

```
Chromosome 1
100 0 100 100 4 5 6 100 100 1 3 22 2 100 101 9 101 101 8 101 101 7 12 11 101 101 10 101 102 13
14 15 102 102 16 17 18 23 102 103 19 20 21 103

Chromosome 2
100 0 100 100 5 6 100 100 1 22 3 2 100 100 4 100 101 10 101 101 9 8 7 12 11 101 102 13 14 15
102 102 16 17 18 23 102 103 19 103 103 20 21 103

Chromosome 3
100 0 2 100 100 5 1 100 100 3 4 6 7 100 101 10 101 101 13 14 16 101 101 9 8 11 12 101 102 23 15
102 102 17 102 103 21 19 22 103 103 20 18 103

Chromosome 4
100 0 2 7 100 100 1 3 4 6 100 101 10 101 101 13 14 16 101 101 8 11 12 101 101 9 5 101 102 15 19
23 102 102 17 102 103 22 21 103 103 18 20 103

Chromosome 5
100 0 4 2 100 100 6 7 17 100 100 1 3 100 101 10 8 101 101 13 14 16 101 101 9 12 5 101 102 15 19
23 102 103 21 22 103 103 11 18 20 103
```

8.6. The genetic algorithm (C++ program)

```
// Genetic.cpp :

#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <stdlib.h>
#include <stdio.h>      /* printf, scanf, puts, NULL */
#include <time.h>
#include <limits.h>
#define MAX_ELITE_PARENTS 70
#define ROUTE_LENGTH 120

using namespace std;

int offspring1[ROUTE_LENGTH];
int testroute[ROUTE_LENGTH];
int routes[400000][ROUTE_LENGTH];
int cost[28][28];
int eliteparents[MAX_ELITE_PARENTS];
int vr[20];          // test partial route

int newroute[ROUTE_LENGTH];
int newroutelen;

int ms[24];
int RoutesN;
```

```

int bestscore,bestscorei;
int biggest;
int mean;
int maxcount;
int wscore=0,wscorei=0;

ofstream genesout("genesout36c.txt",ios::out);

ifstream genes("genes36hc.txt", ios::in);
ifstream cost_file("cost.txt", ios::in);
ifstream timess_file("ms.txt", ios::in);

int fitness(int i);
void crossover(int toc, int p1, int p2);

int locate_elite_parent(int i);
void populate_elite_parents();

void clean_bad_routes(void);
void printroute (int i);
void printroutes (int i, int j);
void findstats(void);
int cprc (int p1, int t1);
void delete_route(int p1);
void load_cost_table(void);
void load_ms_table(void);
void load_genes(void);
void check_twins(int q);
void route_optimization (int p1);

void main(void)
{
    int i,j;

    bestscore=10000;

    load_cost_table();

    load_ms_table();

    load_genes();

    //printroutes(0,RoutesN);

    // best score of initial genes
    for (i=0;i<RoutesN;i++)
    {
        if (bestscore>routes[i][1])
        {
            bestscore=routes[i][1];
            bestscorei=i;
            cout << "New best Score:" << bestscore << " at route " << bestscorei << endl;
        }
    }

    cout <<endl<<"Initial Population :"<<RoutesN<<endl;

    clean_bad_routes();

    cout <<"Population after clean up of bad routes:"<<RoutesN<<endl;

    populate_elite_parents();

    //printout for test purposes
    /*for (j=0;j<RoutesN;j++)
    {
        genesout << j << "\t";
        for (i=0; i<routes[j][0];i++)
            genesout << " " << routes[j][i];
        genesout<<endl;
    }
}

```



```

}
*/

/*****START OF REPOPULATING*****/

int v1, v2, p1, p2, countpop=0;
int toc, ttcupc=0;
bool ttcup=false;
int q;
srand ((int)time(NULL));

while (RoutesN<400000 && countpop++<10000000 )
{
    v1 = rand() % 100;
    v2 = rand() % 100;

    if (v2>80)
    {
        p1=rand() %RoutesN;
        p2=rand() %RoutesN;
    }
    else
    {
        if (v2>45)
        {
            p1=rand() %RoutesN;
            p2=locate_elite_parent(p1);
        }
        else
        {
            p1=locate_elite_parent(-1);
            p2=locate_elite_parent(p1);
        }
    }

    if (routes[p1][1]<2400 )
    {
        if ((v1%2)==0) toc = 1; else toc=3;
    }
    else
    if (routes[p1][1]>2400 )
    {
        toc=2;
    }
    else toc=(v1%3)+1;

    crossover(toc,p1,p2);
    // route_optimization(p1);

    check_twins(RoutesN-1);
    // cout << "___toc:"<<toc<<" p1:"<<p1<<" p2:"<<p2<<endl;

    // printroutes (RoutesN-5,RoutesN);

    // cout <<endl<<"After twins checked"<<endl;
    // printroutes (RoutesN-5,RoutesN);
    // cout << endl<< endl;
    // if (routes[RoutesN-1][1]<2180) printroute (RoutesN-1);

    if (routes[RoutesN-1][2]==0) cout <<"*****ERROR"<<endl<<endl;

    //keep track of best score
    if (bestscore>routes[RoutesN-1][1])

```

```

{
    bestscore=routes[RoutesN-1][1];
    bestscorei=RoutesN-1;
    cout<<endl<<"_____";
    printroute(RoutesN-1);
}

if (RoutesN % 5000==0)
{
    cout <<RoutesN/1000 << "k ";
}

if (RoutesN % 13000==0)
{
    populate_elite_parents();
    for (int t=0;t<MAX_ELITE_PARENTS;t++)
        printroute(eliteparents[t]);
}
// Printout during running
if (RoutesN % 10000==0)
{
    cout <<endl<<RoutesN<<endl;
    cout<<"REPOPULATING ELITE PARENTS - worst score:";
    for (i=0;i<MAX_ELITE_PARENTS;i++)
        printroute(eliteparents[i]);
    for (int d=0;d<RoutesN;d++) check_twins(d);
    populate_elite_parents();
    cout << wscore << " Best Objective So far: " << bestscore << endl;
}

// if ( (RoutesN>20000 && (RoutesN-20000) % 20000 ==0) )
// {
//     cout <<endl<<"Routes before cleaning:"<<RoutesN;
//     for (i=RoutesN-2;i>=0;i--)
//         check_twins(i);
//     cout<<" and after:"<<RoutesN<<endl;
// }

/*
if ( RoutesN>500000 && (RoutesN-500000) % 10000 ==0)
{
    cout <<endl<<"Routes before cleaning bad routes:"<<RoutesN;
    clean_bad_routes();
    cout<<" and after:"<<RoutesN<<endl;
    ttcup=false;
    ttcupc=0;
}
if (ttcup==false && ttcupc++>15000)
    ttcup=true;
*/
}

for (j=0;j<RoutesN;j++)
{
    genesout << j << "\t";
    for (i=0; i<=routes[j][0]-1;i++)
        genesout << " " << routes[j][i];
    genesout<<endl;
}
/*for (i=0;i<28;i++)
{
    for (j=0;j<28;j++)
        cout << " " << cost[i][j];
    cout<<endl;
}
for (i=0;i<24;i++)
    cout << " " << ms[i];
cout <<endl;

```

```

    return 0;
}
*/
}

int fitness(int i)
{
    int j,k;
    int tcost=0, pcost=0,ppcost=0;
    bool vroute=true, troute;

    //  cout << "Fitness of route: "<<i<<endl;
    //  printroute(i);

    for (j=3;j<routes[i][0];j++)
    {
        if (routes[i][j]<100)    // an den epistrefo se depot
        {
            if (routes[i][j-1]<100)    // an den fevgei apo depot
            {
                tcost=tcost+cost[routes[i][j-1]+4][routes[i][j]+4]+ms[routes[i][j]]; // to kostos
                metaksi dio ipostathmon
                pcost=pcost+cost[routes[i][j-1]+4][routes[i][j]+4]+ms[routes[i][j]]; // to kostos
                metaksi dio ipostathmon
            }
            else
            {
                tcost=tcost+cost[routes[i][j-1]-100][routes[i][j]+4]+ms[routes[i][j]]; // an fevgei
                apo depot
                pcost=cost[routes[i][j-1]-100][routes[i][j]+4]+ms[routes[i][j]]; // an fevgei apo
                depot
                vroute=true;
            }
        }
        else    // an eimai se depot
        {
            if (routes[i][j-1]<100) //exo epistrepsei se depot
            {
                tcost=tcost+cost[routes[i][j-1]+4][routes[i][j]-100];
                pcost=pcost+cost[routes[i][j-1]+4][routes[i][j]-100];

                if (pcost>360)
                {
                    vroute=false;
                    ppcost=pcost;
                }
            }
        }
    }
    troute=true;

    /* if (vroute )
    {
        for (k=0;k<RoutesN-1;k++)
            if (routes[k][0]==routes[i][0] && routes[k][1]==routes[i][1])
            {
                for (j=2;j<routes[k][0];j++)
                    if (routes[k][j]!=routes[i][j])
                    {
                        troute=false;
                    }
            }
    }
    */
    if (!vroute || !troute)
    {
        //      cout << "Del:"<<ppcost<<"->v"<<vroute<<" t"<<troute << " "<<endl;
        return -1;
    }
    else

```

```

    return tcost;
}

void crossover(int toc, int p1, int p2)
{
    int t1,t2,i,s1,s2,pcost,k;

    /*      cout <<endl<< "*****Parent1:\t";
       for (i=0;i<parent1[0];i++) cout << parent1[i] << " ";

    */
    int w;

    if (toc==1) // first crossover case
    {
        maxcount=0; //check for infinite do while
        do{
            t1=5 + rand() % (routes[p1][0]-5);

            }while(routes[p1][t1]>=100 && maxcount++<1000);
        if(maxcount>900) cout <<"Error at loop crossover toc1 t1";
        maxcount=0; //check for infinite do while
        do{
            t2=5 + rand() % (routes[p2][0]-5);
            }while(routes[p2][t2]>=100 && maxcount++<1000);
        if(maxcount>900) cout <<"Error at loop crossover toc1 t2";

        s1=routes[p1][t1];
        s2=routes[p2][t2];

        //lets make an offspring
        for (w=0;w<routes[p1][0];w++) routes[RoutesN][w]=routes[p1][w];

        for (i=2;i<routes[RoutesN][0];i++)
        {
            if (routes[RoutesN][i]==s2)
                routes[RoutesN][i]=s1;
            else
                if (routes[RoutesN][i]==s1) routes[RoutesN][i]=s2;
        }
        k=fitness(RoutesN);
        if (k>-1)
        {
            routes[RoutesN][1]=k;
            RoutesN++;
        }

        if (routes[RoutesN-1][routes[RoutesN-1][0]-1]<100)
        {
            cout << "toc= " << toc << " N= " << RoutesN-1 << "->";
            printroute(RoutesN-1);
        }

    }
    else if (toc==2) // stick together two routes
    {
        maxcount=0;
        do
        {
            t1=3+(rand() % (routes[p1][0]-6));

            }while((routes[p1][t1]<100 || routes[p1][t1+1]<100) && maxcount++<1000);
        if(maxcount>900) cout <<"Error at loop crossover toc2 t1";
    }
}

```

```

do
{
    t1--;
}while (routes[p1][t1]<100);

pcost=cprc(p1, t1);

if (pcost<=360)
{
    for (i=0;i<t1;i++)
        routes[RoutesN][i]=routes[p1][i];

    for (i=t1;i<newroutelen+t1;i++) //proto kommati
    {
        routes[RoutesN][i]=newroute[i-t1];
    }

    for (i=t1+newroutelen;i<routes[p1][0];i++) //ipoloipo
    {
        routes[RoutesN][i]=routes[p1][i+2];
    }
    routes[RoutesN][0]=routes[p1][0]-2;
    k=fitness(RoutesN);
    if (k>-1)
    {
        routes[RoutesN][1]=k;
        RoutesN++;
    }

    if (routes[RoutesN-1][routes[RoutesN-1][0]-1]<100)
    {
        cout <<"kollisa 2 !!!! t1="<<t1<<" newlen:"<<newroutelen<< endl<<"From route:";
        printroute(p1);
        cout << " Newroute:" << endl;
        for (i=0;i<newroutelen;i++) cout << newroute[i] << " ";
        cout << endl;
        cout << " Newroute after proto kommati:" << endl;
        for (i=0;i<routes[p1][0];i++) cout << routes[p1][i] << " ";
        cout <<endl;
        cout << "To route:" << endl;
        printroute(p1);

        cout << endl<<endl<<endl<<endl<<endl<<"*****toc= "<< toc<< " N=
"<< RoutesN-1 << "->";
        printroute(RoutesN-1);
    }

}

else if (toc==3 && routes[p1][0]<50) /// break a route
{
    maxcount=0;
    do
    {
        t1=4+ (rand() % (routes[p1][0]-7));
    }while((routes[p1][t1]>=100 || routes[p1][t1+1]>=100) && maxcount++<1000);
    if (maxcount>900) cout << "Error toc3";
    t2=t1;
    do
    {
        t2--;
    }while (routes[p1][t2]<100);
    // cout << "*****time to brake at "<<routes[p1][t1]<< " with substation " <<
    routes[p1][t2]<<endl;
    // printroute(p1);

    routes[p1][0]+=2;

    for (i=routes[p1][0];i>t1;i--)
        routes[RoutesN][i]=routes[p1][i-2];
    routes[RoutesN][t1+1]=routes[RoutesN][t1+2]=routes[p1][t2];

```

```

        for (i=0;i<=t1;i++)
            routes[RoutesN][i]=routes[p1][i];

        routes[p1][0]-=2;
        k=fitness(RoutesN);
        if (k>-1){
            routes[RoutesN][1]=k;
            RoutesN++;
        }

        if (routes[RoutesN-1][routes[RoutesN-1][0]-1]<100)
        {
            cout << "toc= " << toc << " N= " << RoutesN-1 << "->";
            printroute(RoutesN-1);
        }
//    printroute(p1);
    }
}

void printroute (int i)
{
    int j;
    cout <<i<<" ";
    genesout <<i<<" ";

    for (j=0;j<routes[i][0];j++)
    {
        cout <<routes[i][j]<<" ";
        genesout <<routes[i][j]<<" ";
    }
    cout <<endl;
    genesout <<endl;
}

void printroutes (int m, int k)
{
    int j,i;

    for (i=m;i<=k;i++)
    {
        cout <<i<<" ";
        genesout <<i<<" ";
        for (j=0;j<routes[i][0];j++)
        {
            cout <<routes[i][j]<<" ";
            genesout <<routes[i][j]<<" ";
        }
        cout <<endl;
        genesout <<endl;
    }
}

int cprc (int p1, int t1) // Calculate partial route cost 2 routes after routes[p1][t1].
{
    int j, tcost=0, m=0;
    bool first, middle1, middle2, last;
    first=true;
    middle1= true;
    middle2= true;
    last=true;

    for (j=t1;j<routes[p1][0];j++) //make newroute combing two routes
    {
        if (routes[p1][j]>=100 && first)
        {
            first=false;
            newroute[m++]=routes[p1][j];
        }
        else if (routes [p1][j]>=100 && middle1)
        {
            middle1=false;
        }
        else if (routes [p1][j]>=100 && middle2)
        {

```

```

        middle2=false;
    }
    else if (routes [p1][j]>=100 && last)
    {
        last=false;
        newroute[m++]=routes[p1][t1];
    }
    else if (routes[p1][j]<100 && last)
    {
        newroute[m++]=routes[p1][j];
    }
}

for (j=1;j<m;j++)
{
    if (newroute[j]<100)    // an den epistrefo se depot
    {
        if (newroute[j-1]<100) // an den fevgei apo depot
        {
            tcost=tcost+cost[newroute[j-1]+4][newroute[j]+4]+ms[newroute[j]];    // to   kostos
            metaksi dio ipostathmon
        }
        else
        {
            tcost=tcost+cost[newroute[j-1]-100][newroute[j]+4]+ms[newroute[j]];    // an   fevgei
            apo depot
        }
    }
    else    // an eimai se depot
    {
        if (newroute[j-1]<100) //exo epistrepsei se depot
        {
            tcost=tcost+cost[newroute[j-1]+4][newroute[j]-100];
        }
    }
}
newroutelen=m;
return tcost;

}

void clean_bad_routes (void)
{
    for (int i=0;i<RoutesN;i++)
    {
        if (routes[i][1]>5000) // bad chromosome
        {
            delete_route(i);
            i--;
        }
    }
}

void delete_route(int p1)
{
    if (p1!=RoutesN-1)
    {
        for (int w=0;w<routes[RoutesN-1][0];w++)
            routes[p1][w]=routes[RoutesN-1][w];
        RoutesN--;
    }
    else
        RoutesN--;
    //    cout << "Deleted";
}

int locate_elite_parent(int p1)
{
    int k;
    if (p1<0 || p1>RoutesN)

```

```

    pl=-1;
    do
    {
        k=eliteparents[rand() % MAX_ELITE_PARENTS];
    }while (k==pl);
    return k;
}

void populate_elite_parents()
{
    int i,j,k,max,maxi;
    bool ai;

    for (i=0;i<MAX_ELITE_PARENTS;i++) // initial feeding of eliteparents
        eliteparents[i]=i;
    for (i=MAX_ELITE_PARENTS;i<RoutesN;i++) // check the rest routes
    {
        max=0;
        for (j=0;j<MAX_ELITE_PARENTS;j++)
            if (routes[eliteparents[j]][1]>max)
            {
                max=routes[eliteparents[j]][1];
                maxi=j;
            }
        if (routes[i][1]<max)
        {
            eliteparents[maxi]=i;
        }
    }

    wscore=0;
    for (j=0;j<MAX_ELITE_PARENTS;j++)
        if (routes[eliteparents[j]][1]>wscore)
        {
            wscore=routes[eliteparents[j]][1];
            wscorei=j;
        }
}

void load_cost_table(void)
{
    int i=0, j=0;

    //load cost table
    i=0;
    if (cost_file.good())
    {
        string str;

        while(getline(cost_file,str))
        {
            j=0;
            istringstream ss(str);
            int num;

            while(ss >> num && j<28)
            {
                cost[i][j++]=num;
            }
            i++;
        }
    }
}

void load_ms_table(void)
{
    int j=0;

```



```

//load ms table

if (timess_file.good())
{
    string str;

    while(getline(timess_file,str))
    {

        istringstream ss(str);
        int num;

        while(ss >> num && j<28)
        {
            ms[j]=num;
        }
        j++;
    }
}

void load_genes(void)
{
    int j,i,k,e,a,twin=0;
    // load genes
    j=0;
    if (genes.good())
    {
        string str;
        while(getline(genes, str))
        {
            i=2;
            istringstream ss(str);
            int num;
            while(ss >> num)
            {
                routes[j][i++]=num;
            }
            routes[j][0]=i;
            routes[j][1]=fitness(j);
            // if(j>0)
            // {
            //     twin=1;
            //     for (e=0;e<j && twin==1;e++)
            //     {
            //         for(a=0;a<routes[e][0] && twin==1 ;a++)
            //         {
            //             if (routes[e][a]!=routes[j][a]) {
            //                 twin=0;
            //                 printroute(e);
            //                 printroute(j);
            //                 cout << " Differ at :" << a << " element " << routes[e][a] << " " <<
            routes[j][a] << endl;
            //             }
            //         }
            //     }
            // }

            // if (twin==0)
            // {
            //     j++;
            // }

            // cout << j <<" ";

        }
    }
    RoutesN=j;
}

void check_twins(int j)
{
    int i, k,h;
    bool st;

```

```

for (i=0;i<RoutesN-1;i++)
{
    if (routes[i][0]==routes[j][0] && routes[i][1]==routes[j][1])
    {
        st=true;
        for (k=2;k<routes[i][0];k++)
        {
            if (routes[i][k]!=routes[j][k])
                st=false;
        }
        if(st) // found twins!!!!

            RoutesN--;

    }

}

}

void route_optimization (int p1)
{
    int k=2,i, vrlen , initial_cost, bestvalue, bestvaluej,j;
    cout <<endl<<endl<<"Declared Cost: " << routes[p1][1] << endl;;
    do
    {
        i=0;
        vr[i++]=routes[p1][k++];

        do
        {
            vr[i++]=routes[p1][k];
        }while (routes[p1][k++]<100);

        //      for (int z=0;z<i;z++) cout << vr[z] << " ";
        //      cout << " ---> " << i ;

        vrlen=i; //leng of array vr[]

        for (i=0;i<vrlen-1;i++)
        {
            if (i==0) initial_cost=cost[vr[i]-100][vr[i+1]+4] ;
            else if (vr[i+1]>=100)
            {
                initial_cost=initial_cost+cost[vr[i]+4][vr[i+1]-100]+ms[vr[i]];
            }
            else
            {
                initial_cost=initial_cost+cost[vr[i]+4][vr[i+1]+4]+ms[vr[i]];
            }
        }

        //      cout <<" Cost: "<< initial_cost << " " << endl;

        // first check for best depot
        bestvalue=initial_cost;
        bestvaluej=vr[0];
        for (j=100;j<104;j++)
        {
            vr[0]=j;
            vr[vrlen-1]=j;

            for (i=0;i<vrlen-1;i++)
            {
                if (i==0) initial_cost=cost[vr[i]-100][vr[i+1]+4] ;
                else if (vr[i+1]>=100)
                {
                    initial_cost=initial_cost+cost[vr[i]+4][vr[i+1]-100]+ms[vr[i]];
                }
                else
                {
                    initial_cost=initial_cost+cost[vr[i]+4][vr[i+1]+4]+ms[vr[i]];
                }
            }
        }
    }
}

```

```

    }

    if (initial_cost<bestvalue)
    {
//          cout << "With Depot " << bestvaluej << " we had "<<bestvalue<<" but with depot "
<< j <<" we have better value: " << initial_cost<<endl;
        bestvalue=initial_cost;
        bestvaluej=vr[0];

    }
    vr[0]=bestvaluej;
    vr[vrlen-1]=bestvaluej;
}

}while (k<routes[pl][0]);
}

```

8.7. CPLEX output

Cplex optimization was running for several days. An output example is included below:

72H - All Substations

Substations: 24

Tried aggregator 1 time.
MIP Presolve eliminated 40 rows and 1 columns.
MIP Presolve modified 34200 coefficients.
Reduced MIP has 7488 rows, 676 columns, and 34656 nonzeros.
Reduced MIP has 612 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.01 sec. (10.93 ticks)
Probing time = 0.00 sec. (0.71 ticks)
Tried aggregator 1 time.
Presolve time = 0.02 sec. (9.82 ticks)
Probing time = 0.00 sec. (0.71 ticks)
Clique table members: 48.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 6 threads.
Root relaxation solution time = 0.08 sec. (19.67 ticks)

	Nodes					Cuts/		
	Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
	0	0	0.0000	48		0.0000	109	
	0	0	0.0000	96		Cuts: 325	161	
	0	0	0.0000	53		Cuts: 10	168	
*	0+	0			5109.0000	0.0000	235	100.00%
	0	0	0.0000	103	5109.0000	Cuts: 202	235	100.00%
*	0+	0			4578.0000	0.0000	235	100.00%
	0	2	0.0000	61	4578.0000	0.0000	235	100.00%
Elapsed time = 1.14 sec. (498.89 ticks, tree = 0.01 MB, solutions = 2)								
*	3+	3			4318.0000	0.0000	245	100.00%
*	7+	7			3823.0000	0.0000	279	100.00%
	209	211	0.0000	66	3823.0000	0.0000	1230	100.00%
	592	584	1186.0000	41	3823.0000	0.0000	2770	100.00%
	1311	1243	483.0000	48	3823.0000	0.0000	4707	100.00%
*	1446+	1360			3678.0000	0.0000	5040	100.00%
*	1446+	1360			3654.0000	0.0000	5040	100.00%
	1452	1365	0.0000	53	3654.0000	0.0000	5642	100.00%
	1459	1370	0.0000	52	3654.0000	0.0000	5659	100.00%
	1470	1374	211.0000	49	3654.0000	0.0000	5694	100.00%
	1520	1403	182.0000	49	3654.0000	0.0000	5837	100.00%
	1600	1449	329.0000	44	3654.0000	0.0000	6111	100.00%
	1808	1510	1152.0000	31	3654.0000	0.0000	6720	100.00%
*	1998+	1239			3526.0000	0.0000	7280	100.00%
*	1998+	999			3498.0000	0.0000	7280	100.00%
*	1998+	839			3470.0000	0.0000	7280	100.00%
*	1998+	732			3451.0000	0.0000	7280	100.00%

```

* 1998+ 660 3409.0000 0.0000 7280 100.00%
3194 1617 2308.0000 18 3409.0000 0.0000 9984 100.00%
Elapsed time = 13.20 sec. (5013.68 ticks, tree = 1.74 MB, solutions = 12)
6206 4415 2579.0000 15 3409.0000 0.0000 16863 100.00%
10917 8735 2736.0000 18 3409.0000 0.0000 26705 100.00%
15290 12831 652.0000 40 3409.0000 0.0000 35251 100.00%
* 29242+25010 2533.0000 0.0000 61356 100.00%
* 29242+25010 2526.0000 0.0000 61356 100.00%
* 29242+25010 2511.0000 0.0000 61356 100.00%
* 29242+25010 2506.0000 0.0000 61356 100.00%
* 29242+25010 2423.0000 0.0000 61356 100.00%
* 29242+25010 2414.0000 0.0000 61356 100.00%
* 29242+25010 2408.0000 0.0000 61356 100.00%
* 29242+25010 2403.0000 0.0000 61356 100.00%
* 29243+25011 2310.0000 0.0000 61359 100.00%
* 29243+25011 2309.0000 0.0000 61359 100.00%
* 29243+25011 2291.0000 0.0000 61359 100.00%
* 29243+25011 2290.0000 0.0000 61359 100.00%
* 29243+25011 2280.0000 0.0000 61359 100.00%
* 29243+25011 2279.0000 0.0000 61359 100.00%
* 29243+25011 2278.0000 0.0000 61359 100.00%
* 29243+25011 2270.0000 0.0000 61359 100.00%
* 29243+25011 2266.0000 0.0000 61359 100.00%
34280 28436 2235.0000 16 2266.0000 0.0000 70969 100.00%
39610 33449 260.0000 46 2266.0000 0.0000 81110 100.00%
45024 38475 1962.0000 32 2266.0000 0.0000 91003 100.00%
Elapsed time = 48.83 sec. (17198.89 ticks, tree = 49.38 MB, solutions = 162)
50361 43415 1187.0000 34 2266.0000 0.0000 100800 100.00%
55843 48537 1022.0000 36 2266.0000 0.0000 110548 100.00%
61821 54079 2149.0000 25 2266.0000 0.0000 120924 100.00%
67456 59309 989.0000 39 2266.0000 0.0000 130296 100.00%
72967 64424 1809.0000 30 2266.0000 0.0000 140200 100.00%
78650 69762 1268.0000 38 2266.0000 0.0000 150579 100.00%
83490 74263 509.0000 44 2266.0000 0.0000 159109 100.00%
87761 78238 2233.0000 23 2266.0000 0.0000 167105 100.00%
94020 84076 1892.0000 26 2266.0000 0.0000 177971 100.00%
99407 89113 903.0000 39 2266.0000 0.0000 187747 100.00%
Elapsed time = 79.25 sec. (26743.33 ticks, tree = 114.87 MB, solutions = 162)
104109 93491 2211.0000 26 2266.0000 0.0000 196088 100.00%
109310 98331 1453.0000 31 2266.0000 0.0000 204999 100.00%
Began writing nodes to disk (directory c:/cplextemp/cpxa03744 created)
114296 103014 2233.0000 22 2266.0000 0.0000 214009 100.00%
119084 107495 1479.0000 29 2266.0000 0.0000 222163 100.00%
123810 111905 1303.0000 36 2266.0000 0.0000 231022 100.00%
130010 117681 501.0000 44 2266.0000 0.0000 241816 100.00%
134415 121822 1368.0000 30 2266.0000 0.0000 250331 100.00%
140446 127472 1520.4735 36 2266.0000 0.0000 260753 100.00%
145266 131982 445.0000 44 2266.0000 0.0000 269215 100.00%
150886 137245 1741.0000 32 2266.0000 0.0000 279127 100.00%
Elapsed time = 108.83 sec. (36327.45 ticks, tree = 176.92 MB, solutions = 162)
Nodefile size = 48.97 MB (5.87 MB after compression)
*151195+137538 2263.0000 0.0000 279635 100.00%
*151195+137538 2260.0000 0.0000 279635 100.00%
*151426+137753 2254.0000 0.0000 279996 100.00%
155434 141504 1219.0000 35 2254.0000 0.0000 287403 100.00%
*159598+145416 2249.0000 0.0000 294245 100.00%
160543 146297 1974.0000 30 2249.0000 0.0000 295784 100.00%
165393 150836 1142.0000 37 2249.0000 0.0000 303936 100.00%
170473 155604 531.0000 42 2249.0000 0.0000 312722 100.00%
*172479+157467 2241.0000 0.0000 316188 100.00%
175202 160034 204.0000 46 2241.0000 0.0000 320584 100.00%
180565 165103 2166.0000 23 2241.0000 0.0000 330539 100.00%
185209 169461 1654.0000 33 2241.0000 0.0000 338810 100.00%
190721 174609 212.4735 44 2241.0000 0.0000 349184 100.00%
195405 178950 1922.0000 28 2241.0000 0.0000 357878 100.00%
200040 183271 1449.0000 32 2241.0000 0.0000 366113 100.00%
Elapsed time = 142.05 sec. (45968.73 ticks, tree = 235.13 MB, solutions = 169)
Nodefile size = 106.93 MB (12.18 MB after compression)
204483 187337 2040.0000 30 2241.0000 0.0000 373699 100.00%
209495 191943 779.0000 40 2241.0000 0.0000 382085 100.00%
214136 196169 927.0000 38 2241.0000 0.0000 389694 100.00%

2 hours later.....

Elapsed time = 8393.76 sec. (1892317.12 ticks, tree = 11490.86 MB, solutions = 184)
Nodefile size = 11362.79 MB (1026.11 MB after compression)

```

9714741	8991965	1953.0000	32	2034.0000	2.8800	20033430	99.86%
9734281	9009784	cutoff		2034.0000	2.8800	20069749	99.86%
9753940	9027833	233.2407	57	2034.0000	2.8800	20105143	99.86%
9773801	9046171	1426.8182	45	2034.0000	2.8800	20141381	99.86%
9793309	9064378	552.8800	54	2034.0000	2.8800	20178174	99.86%
9812623	9082127	607.2407	50	2034.0000	2.8800	20212273	99.86%
9832133	9100163	1486.0000	40	2034.0000	2.8800	20248801	99.86%
9852720	9119131	602.8800	50	2034.0000	2.8800	20288597	99.86%
9872326	9137077	375.2407	56	2034.0000	2.8800	20326227	99.86%
9892364	9155446	1789.0000	32	2034.0000	2.8800	20365273	99.86%
Elapsed time = 8563.69 sec. (1930632.80 ticks, tree = 11724.54 MB, solutions = 184)							
Nodefile size = 11596.65 MB (1046.50 MB after compression)							
*9893562+9156569				2007.0000	2.8800	20367547	99.86%
9910714	9170630	117.0000	58	2007.0000	2.8800	20401260	99.86%
9930947	9189336	540.8840	54	2007.0000	2.8800	20440712	99.86%
9950650	9207316	231.8800	61	2007.0000	2.8800	20480421	99.86%
9969325	9224342	688.8800	46	2007.0000	2.8800	20516703	99.86%
9988197	9241500	1800.0000	36	2007.0000	2.8800	20553836	99.86%
30 hours later.....							
Elapsed time = 115491.33 sec. (27206318.05 ticks, tree = 159148.20 MB, solutions = 185)							
Nodefile size = 159020.67 MB (14862.39 MB after compression)							
138114598	123905118	1772.0000	33	2007.0000	7.1849	2.99e+008	99.64%
138194724	123978855	1187.0000	43	2007.0000	7.1849	2.99e+008	99.64%
138274170	124051276	1214.0000	42	2007.0000	7.1849	2.99e+008	99.64%
138353203	124122566	1206.8800	44	2007.0000	7.1849	3.00e+008	99.64%
138431448	124191264	1381.1849	41	2007.0000	7.1849	3.00e+008	99.64%
138511030	124261710	497.8800	56	2007.0000	7.1849	3.00e+008	99.64%
138590642	124332150	1190.8800	45	2007.0000	7.1849	3.00e+008	99.64%
138669575	124403148	846.1849	47	2007.0000	7.1849	3.00e+008	99.64%
138748445	124472660	1306.0000	48	2007.0000	7.1849	3.00e+008	99.64%
138828032	124544595	1116.3050	41	2007.0000	7.1849	3.01e+008	99.64%
Elapsed time = 116136.54 sec. (27358922.96 ticks, tree = 160065.11 MB, solutions = 185)							
Nodefile size = 159937.07 MB (14945.57 MB after compression)							
138909208	124619151	1274.0000	43	2007.0000	7.1849	3.01e+008	99.64%
138989667	124693287	1835.3050	39	2007.0000	7.1849	3.01e+008	99.64%
139070785	124767798	1166.0000	48	2007.0000	7.1849	3.01e+008	99.64%
139151287	124841030	749.3050	56	2007.0000	7.1849	3.01e+008	99.64%
139231133	124914607	724.6114	52	2007.0000	7.1849	3.01e+008	99.64%
139308276	124985086	525.5450	55	2007.0000	7.1849	3.02e+008	99.64%
139387789	125055741	283.1136	56	2007.0000	7.1849	3.02e+008	99.64%
*139462442+125123323				1991.0000	7.1849	3.02e+008	99.64%
139463272	125124086	196.1849	64	1991.0000	7.1849	3.02e+008	99.64%
*139467172+125127732				1989.0000	7.1849	3.02e+008	99.64%
*139470613+125130107				1988.0000	7.1849	3.02e+008	99.64%
*139484156+125142573				1987.0000	7.1849	3.02e+008	99.64%
139521981	125176964	1592.0000	39	1987.0000	7.1849	3.02e+008	99.64%
139581536	125230913	1110.6781	49	1987.0000	7.1871	3.02e+008	99.64%
Elapsed time = 116779.57 sec. (27511594.83 ticks, tree = 160947.29 MB, solutions = 191)							
Nodefile size = 160819.51 MB (15024.71 MB after compression)							
139637739	125281397	444.1871	56	1987.0000	7.1871	3.02e+008	99.64%
139695656	125332675	1342.0435	44	1987.0000	7.1906	3.03e+008	99.64%
139753456	125384459	1300.0000	44	1987.0000	7.1946	3.03e+008	99.64%
Best solution of this scenario after 72 hours was 1987 objective value							

8.8. Generation of scenarios

Code that produces pseudorandom distance matrices.

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>

void main()
{
    FILE *fr,*fw,*fn;
    int i, j, k=0,u ;
    int temp;
    int km[29][29];
```

```

char name[32][30];

srand( (unsigned)time( NULL ) );

//OPEN FILES
fr=fopen("apostasiologio.txt", "r");
if (fr==NULL) {
    printf("Cannot open file apostasiologio.txt\nExiting.....\n");
    exit(1);
}
fn=fopen("substations.txt", "r");
if (fn==NULL) {
    printf("Cannot open file substations.txt\nExiting.....\n");
    exit(1);
}
fw=fopen("senariol.txt", "w+");
if (fr==NULL) {
    printf("Cannot open file distances.txt\nExiting.....\n");
    exit(1);
}

// READ SUBSTATION NAMES save them to name[i]
for(i=0;i<32;i++){
    fgets(name[i],30,fn);
}

// READ KM save them to km[32][32]
for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        fscanf(fr,"%i",&temp);
        km[i][j] = temp;
    }
}

// PRINT EVERYTHING
for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        printf("%i ",km[i][j]);
    }
    printf("\n");
}
for(i=0;i<29;i++){
    printf("%s",name[i]);
}

for (i=0;i<29;i++){
    for (j=0;j<29;j++){
        if (i!=j) u=rand()%100-25; else u=0;
        km[i][j]=km[i][j]+u;
    }
}

for(i=0;i<29;i++){
    for(j=i;j<29;j++){
        if (km[i][j]>km[j][i]){
            km[i][j]=km[j][i];
            k++;
        }
        if (km[i][j]<km[j][i])
        {
            km[j][i]=km[i][j];
            k++;
        }
    }
}

```

```

printf("/n/n ALLAGES=%i \n",k);

for(i=0;i<29;i++){
    for(j=0;j<29;j++){
        fprintf(fw,"%i\t",abs(km[i][j]));
        printf("%4i ",abs(km[i][j]));
    }
    fprintf(fw,"\n");
    printf("\n");
}

}

```

8.9. Scenarios

Apart from the real data used (scenario 0) the following data were fed as distances in minutes to the algorithms:

Scenario 1

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	107	112	24	27	8	24	38	19	29	39	79	24	23	110	64	53	59	91	108	45	127	83	53	29	111	9	44
d1	107	0	64	104	57	50	58	124	61	56	91	7	64	65	44	53	54	55	50	111	96	84	102	113	91	145	88	80
d2	112	64	0	102	69	101	88	114	97	126	95	62	128	75	132	53	56	53	44	45	41	51	87	73	115	120	103	70
d3	24	104	102	0	60	78	41	55	52	80	88	114	133	68	139	148	105	112	90	141	87	85	91	21	10	95	36	54
s0	27	57	69	60	0	8	34	25	2	53	25	45	35	84	102	97	41	54	119	125	66	61	57	123	92	83	37	65
s1	8	50	101	78	8	0	46	74	23	98	50	57	102	77	147	82	93	111	115	114	113	100	77	89	49	129	42	56
s2	24	58	88	41	34	46	0	51	24	33	42	71	37	23	83	34	80	115	78	106	68	66	82	106	86	96	31	64
s3	38	124	114	55	25	74	51	0	33	74	131	91	88	84	150	97	105	109	114	159	104	110	93	83	82	82	42	100
s4	19	61	97	52	2	23	24	33	0	35	46	29	71	114	116	92	39	99	140	121	111	85	106	153	95	92	51	56
s5	29	56	126	80	53	98	33	74	35	0	49	83	100	79	159	115	139	143	113	161	86	115	141	146	81	114	120	87
s6	39	91	95	88	25	50	42	131	46	49	0	36	96	123	104	121	87	67	95	155	126	82	120	88	128	133	49	61
s7	79	7	62	114	45	57	71	91	29	83	36	0	86	40	89	85	57	24	64	87	56	99	147	91	100	86	61	47
s8	24	64	128	133	35	102	37	88	71	100	96	86	0	10	61	47	58	31	63	106	150	93	69	124	87	90	117	95
s9	23	65	75	68	84	77	23	84	114	79	123	40	10	0	84	16	22	117	119	117	73	110	77	147	162	119	60	48
s10	110	44	132	139	102	147	83	150	116	159	104	89	61	84	0	87	125	137	122	179	158	147	180	165	141	232	170	162
s11	64	53	53	148	97	82	34	97	92	115	121	85	47	16	87	0	45	69	65	105	92	116	71	85	84	111	65	59
s12	53	54	56	105	41	93	80	105	39	139	87	57	58	22	125	45	0	19	80	73	78	60	57	83	84	113	105	65
s13	59	55	53	112	54	111	115	109	99	143	67	24	31	117	137	69	19	0	65	93	44	95	95	128	92	165	125	103
s14	91	50	44	90	119	115	78	114	140	113	95	64	63	119	122	65	80	65	0	91	35	60	67	154	117	134	149	54
s15	108	111	45	141	125	114	106	159	121	161	155	87	106	117	179	105	73	93	91	0	58	77	73	66	144	175	119	109
s16	45	96	41	87	66	113	68	104	111	86	126	56	150	73	158	92	78	44	35	58	0	4	27	33	118	99	95	16
s17	127	84	51	85	61	100	66	110	85	115	82	99	93	110	147	116	60	95	60	77	4	0	58	42	70	123	53	44
s18	83	102	87	91	57	77	82	93	106	141	120	147	69	77	180	71	57	95	67	73	27	58	0	20	94	117	107	29
s19	53	113	73	21	123	89	106	83	153	146	88	91	124	147	165	85	83	128	154	66	33	42	20	0	97	65	78	43
s20	29	91	115	10	92	49	86	82	95	81	128	100	87	162	141	84	84	92	117	144	118	70	94	97	0	82	72	75
s21	111	145	120	95	83	129	96	82	92	114	133	86	90	119	232	111	113	165	134	175	99	123	117	65	82	0	96	100
s22	9	88	103	36	37	42	31	42	51	120	49	61	117	60	170	65	105	125	149	119	95	53	107	78	72	96	0	36
s23	44	80	70	54	65	56	64	100	56	87	61	47	95	48	162	59	65	103	54	109	16	44	29	43	75	100	36	0

Scenario 2

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	61	77	53	12	52	10	65	91	69	98	21	29	91	152	94	56	89	92	121	41	76	72	78	35	96	55	35
d1	61	0	102	73	58	81	62	71	87	70	55	55	7	13	109	70	36	36	134	120	86	121	79	127	116	100	55	86
d2	77	102	0	100	159	143	59	158	87	156	109	97	108	65	128	98	64	61	9	72	70	59	52	53	92	103	90	47
d3	53	73	100	0	55	83	45	127	95	112	74	97	102	106	131	120	92	112	118	135	139	123	107	17	39	5	32	46
s0	12	58	159	55	0	6	24	28	30	22	67	7	24	63	133	63	32	137	120	124	69	108	88	63	53	116	67	24
s1	52	81	143	83	6	0	20	89	47	39	59	18	47	44	111	47	71	77	118	122	70	71	151	104	64	72	69	102
s2	10	62	59	45	24	20	0	91	91	54	59	18	35	64	95	34	34	62	71	100	79	97	97	50	113	131	22	11
s3	65	71	158	127	28	89	91	0	116	91	66	118	138	64	179	114	126	95	169	161	116	119	127	89	114	123	90	109
s4	91	87	87	95	30	47	91	116	0	37	53	21	49	53	116	58	47	101	81	155	137	68	65	75	64	86	68	32
s5	69	70	156	112	22	39	54	91	37	0	64	45	52	59	128	123	77	104	103	180	130	92	91	113	89	107	128	138
s6	98	55	109	74	67	59	59	66	53	64	0	28	55	123	123	118	65	114	97	179	71	116	94	123	75	103	91	71
s7	21	55	97	97	7	18	18	118	21	45	28	0	0	1	96	43	32	71	97	81	98	69	107	134	55	86	65	48
s8	29	7	108	102	24	47	35	138	49	52	55	0	0	41	76	106	64	68	51	102	100	84	129	132	74	127	107	78
s9	91	13	65	106	63	44	64	64	53	59	123	1	41	0	77	90	32	83	93	91	132	74	100	103	134	167	83	43
s10	152	109	128	131	133	111	95	179	116	128	123	96	76	77	0	108	86	134	174	159	197	144	174	151	153	165	168	103
s11	94	70	98	120	63	47	34	114	58	123	118	43	106	90	108	0	5	42	48	131	67	100	137	105	129	175	70	40

s12	56	36	64	92	32	71	34	126	47	77	65	32	64	32	86	5	0	50	57	65	82	102	79	84	94	153	105	40
s13	89	36	61	112	137	77	62	95	101	104	114	71	68	83	134	42	50	0	47	94	54	57	122	140	150	144	113	53
s14	92	134	9	118	120	118	71	169	81	103	97	97	51	93	174	48	57	47	0	58	39	97	125	84	118	151	103	88
s15	121	120	72	135	124	122	100	161	155	180	179	81	102	91	159	131	65	94	58	0	125	52	118	68	104	145	138	70
s16	41	86	70	139	69	70	79	116	137	130	71	98	100	132	197	67	82	54	39	125	0	77	30	60	49	89	67	20
s17	76	121	59	123	108	71	97	119	68	92	116	69	84	74	144	100	102	57	97	52	77	0	27	37	59	94	74	27
s18	72	79	52	107	88	151	97	127	65	91	94	107	129	100	174	137	79	122	125	118	30	27	0	103	81	65	48	60
s19	78	127	53	17	63	104	50	89	75	113	123	134	132	103	151	105	84	140	84	68	60	37	103	0	19	51	60	108
s20	35	116	92	39	53	64	113	114	64	89	75	55	74	134	153	129	94	150	118	104	49	59	81	19	0	19	47	45
s21	96	100	103	5	116	72	131	123	86	107	103	86	127	167	165	175	153	144	151	145	89	94	65	51	19	0	59	114
s22	55	55	90	32	67	69	22	90	68	128	91	65	107	83	168	70	105	113	103	138	67	74	48	60	47	59	0	83
s23	35	86	47	46	24	102	11	109	32	138	71	48	78	43	103	40	40	53	88	70	20	27	60	108	45	114	83	0

Scenario 3

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	40	69	60	26	74	47	52	17	102	59	55	84	33	87	128	64	70	79	97	59	66	60	71	57	68	24	39
d1	40	0	64	74	77	56	58	103	67	113	69	13	35	34	55	76	44	53	125	116	97	104	101	120	126	156	80	76
d2	69	64	0	77	83	104	56	184	116	120	94	97	132	125	212	68	84	20	13	61	42	28	58	95	68	139	78	62
d3	60	74	77	0	95	54	61	82	115	98	112	67	139	71	154	125	122	107	89	126	105	48	77	75	3	11	31	88
s0	26	77	83	95	0	69	6	92	19	43	28	19	81	94	155	53	68	68	66	118	84	97	117	55	122	148	73	83
s1	74	56	104	54	69	0	2	74	39	72	88	38	86	112	135	94	92	79	85	123	71	72	108	71	86	68	109	85
s2	47	58	56	61	6	2	0	46	59	88	49	66	95	79	131	90	62	104	110	160	86	100	112	52	101	70	61	76
s3	52	103	184	82	92	74	46	0	92	64	64	66	101	100	144	76	154	124	146	176	162	95	120	97	78	116	53	69
s4	17	67	116	115	19	39	59	92	0	23	28	19	88	34	115	61	65	56	98	114	69	111	125	86	50	84	56	55
s5	102	113	120	98	43	72	88	64	23	0	68	72	54	72	156	79	97	95	97	175	111	177	104	155	113	155	96	97
s6	59	69	94	112	28	88	49	64	28	68	0	27	49	37	176	79	57	89	115	130	80	122	152	117	118	97	69	45
s7	55	13	97	67	19	38	66	66	19	72	27	0	4	9	64	20	88	35	50	100	104	144	63	132	124	75	70	29
s8	84	35	132	139	81	86	95	101	88	54	49	4	0	18	91	110	34	62	82	141	118	122	109	147	73	175	88	91
s9	33	34	125	71	94	112	79	100	34	72	37	9	18	0	60	53	37	70	51	157	101	144	104	90	123	91	61	88
s10	87	55	212	154	155	135	131	144	115	156	176	64	91	60	0	91	90	128	145	152	136	134	188	164	131	203	159	113
s11	128	76	68	125	53	94	90	76	61	79	79	20	110	53	91	0	23	68	114	71	108	106	72	107	170	135	82	68
s12	64	44	84	122	68	92	62	154	65	97	57	88	34	37	90	23	0	71	29	89	71	71	80	87	99	109	79	64
s13	70	53	20	107	68	79	104	124	56	95	89	35	62	70	128	68	71	0	43	76	38	97	69	113	105	146	150	107
s14	79	125	13	89	66	85	110	146	98	97	115	50	82	51	145	114	29	43	0	48	25	82	58	94	121	182	117	74
s15	97	116	61	126	118	123	160	176	114	175	130	100	141	157	152	71	89	76	48	0	54	58	110	99	103	176	136	76
s16	59	97	42	105	84	71	86	162	69	111	80	104	118	101	136	108	71	38	25	54	0	23	20	31	62	72	65	14
s17	66	104	28	48	97	72	100	95	111	177	122	144	122	144	134	106	71	97	82	58	23	0	14	20	45	139	63	56
s18	60	101	58	77	117	108	112	120	125	104	152	63	109	104	188	72	80	69	58	110	20	14	0	18	46	80	48	69
s19	71	120	95	75	55	71	52	97	86	155	117	132	147	90	164	107	87	113	94	99	31	20	18	0	33	68	65	96
s20	57	126	68	3	122	86	101	78	50	113	118	124	73	123	131	170	99	105	121	103	62	45	46	33	0	53	57	70
s21	68	156	139	11	148	68	70	116	84	155	97	75	175	91	203	135	109	146	182	176	72	139	80	68	53	0	63	88
s22	24	80	78	31	73	109	61	53	56	96	69	70	88	61	159	82	79	150	117	136	65	63	48	65	57	63	0	28
s23	39	76	62	88	83	85	76	69	55	97	45	29	91	88	113	68	64	107	74	76	14	56	69	96	70	88	28	0

Scenario 4

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	26	106	64	38	30	11	60	45	85	63	28	24	60	99	67	87	60	120	114	82	65	75	66	53	110	42	78
d1	26	0	79	156	98	58	32	114	61	58	121	23	43	27	57	44	29	75	112	175	98	92	95	144	75	115	77	78
d2	106	79	0	96	86	83	87	168	80	161	100	88	87	81	176	60	68	25	31	67	30	79	72	64	77	118	115	77
d3	64	156	96	0	33	40	122	58	54	104	68	82	70	112	180	117	74	104	88	153	115	75	50	29	2	39	36	72
s0	38	98	86	33	0	51	41	29	27	85	41	12	63	40	140	113	89	97	77	152	59	96	55	65	78	137	56	46
s1	30	58	83	40	51	0	44	90	57	49	55	25	78	75	123	49	59	134	79	155	71	113	72	108	127	82	24	86
s2	11	32	87	122	41	44	0	50	68	79	68	15	47	47	147	48	85	59	128	99	92	62	82	75	43	112	57	44
s3	60	114	168	58	29	90	50	0	68	73	57	93	75	147	179	87	82	128	135	180	131	156	98	108	56	119	53	65
s4	45	61	80	54	27	57	68	68	0	81	55	33	62	33	173	109	52	73	106	114	60	111	69	129	71	134	96	62
s5	85	58	161	104	85	49	79	73	81	0	102	74	114	77	120	121	85	150	122	162	139	119	151	102	170	163	65	98
s6	63	121	100	68	41	55	68	57	55	102	0	35	39	111	101	52	72	119	97	148	95	96	134	108	131	119	108	57
s7	28	23	88	82	12	25	15	93	33	74	35	0	70	10	153	73	42	94	48	124	75	74	85	73	72	102	84	80
s8	24	43	87	70	63	78	47	75	62	114	39	70	0	0	67	66	29	54	116	136	85	77	97	101	87	123	66	90
s9	60	27	81	112	40	75	47	147	33	77	111	10	0	0	100	92	53	38	91	103	97	105	146	110	70	130	60	104
s10	99	57	176	180	140	123	147	179	173	120	101	153	67	100	0	83	77	130	113	181	203	153	142	160	159	211	168	143
s11	67	44	60	117	113	49	48	87	109	121	52	73	66	92	83	0	7	31	86	106	84	81	154	145	106	132	127	90
s12	87	29	68	74	89	59	85	82	52	85	72	42	29	53	77	7	0	24	61	71	79	126	78	150	99	110	84	105
s13	60	75	25	104	97	134	59	128	73	150	119	94	54	38	130	31	24	0	1	47	73	73	117	128	106	174	170	136
s14	120	112	31	88	77	79	128	135	106	122	97	48	116	91	113	86	61	1	0	26	19	63	49	114	134	132	108	94
s15	114	175	67	153	152	155	99	180	114	162	148	124	136	103	181	106	71	47	26	0	56	131	132	63	127	125	114	99
s16	82	98	30	115	59	71	92	131	60	139	95	75	85	97	203	84	79	73	19	56	0	10	22	53	44	122	61	48
s17	65	92	79	75	96	113	62	156	111	119	96	74	77	105	153	81	126	73	63	131	10	0	13	40	69	105	117	61
s18	75	95	72	50	55	72	82	98	69	151	134	85	97	146	142	154	78	117	49	132	22	13	0	59	79	92	46	29
s19	66	144	64	29	65	108	75	108	129	102	108	73	101	110	160	145	150	128	114	63	53	40	59	0	69	69	81	49
s20	53	75	77	2	78	127	43	56	71	170	131	72	87	70	159	106	99	106	134	127	44	69	79	69	0	94	53	107

s21	110	115	118	39	137	82	112	119	134	163	119	102	123	130	211	132	110	174	132	125	122	105	92	69	94	0	51	142
s22	42	77	115	36	56	24	57	53	96	65	108	84	66	60	168	127	84	170	108	114	61	117	46	81	53	51	0	54
s23	78	78	77	72	46	86	44	65	62	98	57	80	90	104	143	90	105	136	94	99	48	61	29	49	107	142	54	0

Scenario 5

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	46	84	58	18	0	34	28	21	33	88	40	49	78	173	48	70	126	96	124	90	107	92	43	97	108	14	28
d1	46	0	73	127	49	40	99	118	121	92	74	52	23	51	107	30	36	63	95	126	122	121	76	96	94	135	101	81
d2	84	73	0	85	72	122	70	141	120	149	141	73	108	74	139	52	96	31	15	76	4	49	58	101	113	108	96	43
d3	58	127	85	0	43	67	54	52	93	98	134	117	95	88	151	96	138	113	112	152	82	74	65	51	23	10	33	47
s0	18	49	72	43	0	1	1	73	7	52	39	66	26	22	103	35	69	136	80	106	135	76	112	110	64	148	70	17
s1	0	40	122	67	1	0	63	80	38	90	103	51	49	86	125	121	100	92	90	122	81	99	94	95	96	74	55	42
s2	34	99	70	54	1	63	0	107	22	46	31	29	44	47	126	37	58	43	86	106	84	80	96	119	68	142	52	22
s3	28	118	141	52	73	80	107	0	50	97	53	54	103	83	156	109	83	114	122	156	160	155	114	105	96	76	72	77
s4	21	121	120	93	7	38	22	50	0	61	58	19	96	77	110	44	66	83	81	142	104	77	72	99	124	108	84	98
s5	33	92	149	98	52	90	46	97	61	0	49	47	99	53	129	92	88	93	106	176	130	120	93	134	153	96	133	92
s6	88	74	141	134	39	103	31	53	58	49	0	79	45	58	141	61	68	114	113	154	80	115	140	136	121	96	97	86
s7	40	52	73	117	66	51	29	54	19	47	79	0	66	41	77	54	54	48	117	134	102	61	75	86	103	136	77	87
s8	49	23	108	95	26	49	44	103	96	99	45	66	0	36	84	22	85	95	91	113	98	134	82	105	108	98	121	108
s9	78	51	74	88	22	86	47	83	77	53	58	41	36	0	64	72	21	59	109	146	109	127	94	82	101	104	58	43
s10	173	107	139	151	103	125	126	156	110	129	141	77	84	64	0	87	92	121	117	160	132	169	154	153	144	165	181	105
s11	48	30	52	96	35	121	37	109	44	92	61	54	22	72	87	0	1	29	47	82	90	109	128	90	109	125	104	42
s12	70	36	96	138	69	100	58	83	66	88	68	54	85	21	92	1	0	54	58	142	61	85	123	104	85	141	110	87
s13	126	63	31	113	136	92	43	114	83	93	114	48	95	59	121	29	54	0	7	96	72	54	102	94	126	178	137	71
s14	96	95	15	112	80	90	86	122	81	106	113	117	91	109	117	47	58	7	0	23	82	74	114	115	124	106	132	59
s15	124	126	76	152	106	122	106	156	142	176	154	134	113	146	160	82	142	96	23	0	57	103	151	75	105	157	190	89
s16	90	122	4	82	135	81	84	160	104	130	80	102	98	109	132	90	61	72	82	57	0	49	18	29	52	118	116	95
s17	107	121	49	74	76	99	80	155	77	120	115	61	134	127	169	109	85	54	74	103	49	0	35	19	43	92	67	61
s18	92	76	58	65	112	94	96	114	72	93	140	75	82	94	154	128	123	102	114	151	18	35	0	88	32	131	76	24
s19	43	96	101	51	110	95	119	105	99	134	136	86	105	82	153	90	104	94	115	75	29	19	88	0	33	80	87	60
s20	97	94	113	23	64	96	68	96	124	153	121	103	108	101	144	109	85	126	124	105	52	43	32	33	0	54	51	55
s21	108	135	108	10	148	74	142	76	108	96	96	136	98	104	165	125	141	178	106	157	118	92	131	80	54	0	68	82
s22	14	101	96	33	70	55	52	72	84	133	97	77	121	58	181	104	110	137	132	190	116	67	76	87	51	68	0	45
s23	28	81	43	47	17	42	22	77	98	92	86	87	108	43	105	42	87	71	59	89	95	61	24	60	55	82	45	0

Scenario 6

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	65	110	52	26	50	15	85	26	49	68	53	42	39	117	64	52	96	75	101	66	56	42	53	46	56	33	78
d1	65	0	68	99	74	90	71	111	74	127	57	58	5	43	100	74	52	62	78	161	73	107	125	155	90	151	63	58
d2	110	68	0	78	85	93	101	112	137	117	119	121	58	135	194	64	56	37	11	105	26	43	44	119	81	167	77	81
d3	52	99	78	0	53	61	87	77	90	92	100	120	89	104	223	150	136	117	105	131	48	54	38	82	14	10	30	53
s0	26	74	85	53	0	20	14	74	78	25	31	20	20	24	131	89	42	138	76	136	80	89	69	133	54	91	68	27
s1	50	90	93	61	20	0	32	37	10	99	56	69	110	55	125	57	52	67	102	166	60	89	90	111	93	74	26	58
s2	15	71	101	87	14	32	0	103	80	60	26	75	31	91	105	31	63	47	94	113	116	58	47	63	68	70	86	68
s3	85	111	112	77	74	37	103	0	107	116	97	77	103	86	195	163	116	100	182	173	110	147	102	104	142	95	61	95
s4	26	74	137	90	78	10	80	107	0	42	61	37	66	38	144	44	81	59	72	171	114	131	107	101	89	113	62	51
s5	49	127	117	92	25	99	60	116	42	0	50	39	85	61	127	119	122	135	113	163	89	120	111	130	82	117	68	86
s6	68	57	119	100	31	56	26	97	61	50	0	25	55	71	107	47	63	108	161	165	96	120	111	95	111	102	94	100
s7	53	58	121	120	20	69	75	77	37	39	25	0	49	4	120	35	21	49	86	84	103	125	118	100	103	139	66	52
s8	42	5	58	89	20	110	31	103	66	85	55	49	0	23	113	34	40	41	80	111	72	92	94	109	91	97	75	94
s9	39	43	135	104	24	55	91	86	38	61	71	4	23	0	116	16	57	61	75	87	103	117	73	128	88	110	119	65
s10	117	100	194	223	131	125	105	195	144	127	107	120	113	116	0	129	96	179	141	169	148	185	152	223	142	183	131	104
s11	64	74	64	150	89	57	31	163	44	119	47	35	34	16	129	0	57	58	86	86	62	124	67	116	81	150	77	61
s12	52	52	56	136	42	52	63	116	81	122	63	21	40	57	96	57	0	53	87	82	56	140	100	97	171	122	105	70
s13	96	62	37	117	138	67	47	100	59	135	108	49	41	61	179	58	53	0	3	92	82	77	76	159	143	197	96	75
s14	75	78	11	105	76	102	94	182	72	113	161	86	80	75	141	86	87	3	0	100	51	43	48	115	82	148	99	99
s15	101	161	105	131	136	166	113	173	171	163	165	84	111	87	169	86	82	92	100	0	107	107	68	87	151	156	129	121
s16	66	73	26	48	80	60	116	110	114	89	96	103	72	103	148	62	56	82	51	107	0	77	31	36	82	114	71	24
s17	56	107	43	54	89	89	58	147	131	120	120	125	92	117	185	124	140	77	43	107	77	0	25	96	77	84	94	61
s18	42	125	44	38	69	90	47	102	107	111	111	118	94	73	152	67	100	76	48	68	31	25	0	18	57	76	127	51
s19	53	155	119	82	133	111	63	104	101	130	95	100	109	128	223	116	97	159	115	87	36	96	18	0	51	36	93	39
s20	46	90	81	14	54	93	68	142	89	82	111	103	91	88	142	81	171	143	82	151	82	77	57	51	0	35	37	78
s21	56	151	167	10	91	74	70	95	113	117	102	139	97	110	183	150	122	197	148	156	114	84	76	36	35	0	54	123
s22	33	63	77	30	68	26	86	61	62	68	94	66	75	119	131	77	105	96	99	129	71	94	127	93	37	54	0	46
s23	78	58	81	53	27	58	68	95	51	86	100	52	94	65	104	61	70	75	99	121	24	61	51	39	78	123	46	0

d0	0	85	93	66	15	3	64	31	30	65	20	35	67	101	121	64	35	64	116	111	42	68	53	117	92	60	52	28
d1	85	0	93	75	50	62	48	87	114	93	108	32	11	67	92	105	71	60	93	129	82	81	73	93	117	94	58	89
d2	93	93	0	148	112	132	64	152	97	158	97	112	117	116	138	66	83	8	2	25	69	55	130	63	74	154	110	80
d3	66	75	148	0	64	87	81	74	71	77	67	64	105	66	194	94	134	162	97	111	55	66	110	107	11	18	62	59
s0	15	50	112	64	0	4	27	31	4	28	82	65	98	39	126	82	34	82	70	139	56	85	57	117	66	102	37	37
s1	3	62	132	87	4	0	65	26	41	40	41	56	59	79	169	64	82	107	113	152	91	90	73	72	68	115	64	62
s2	64	48	64	81	27	65	0	58	4	51	89	20	38	71	84	103	35	111	102	119	37	74	80	107	52	96	105	14
s3	31	87	152	74	31	26	58	0	31	131	55	88	113	87	134	75	117	131	121	236	91	132	121	101	85	105	89	68
s4	30	114	97	71	4	41	4	31	0	22	64	55	36	62	162	68	95	136	90	131	73	107	77	72	138	86	41	35
s5	65	93	158	77	28	40	51	131	22	0	40	89	57	75	125	112	63	110	142	146	92	99	108	158	104	115	68	68
s6	20	108	97	67	82	41	89	55	64	40	0	32	125	85	108	48	59	70	97	177	81	118	83	90	69	108	59	50
s7	35	32	112	64	65	56	20	88	55	89	32	0	4	35	120	16	56	55	61	130	54	105	119	81	111	136	35	44
s8	67	11	117	105	98	59	38	113	36	57	125	4	0	9	73	76	11	55	76	114	113	117	75	110	115	96	88	112
s9	101	67	116	66	39	79	71	87	62	75	85	35	9	0	71	81	92	46	108	161	134	79	121	122	87	116	71	86
s10	121	92	138	194	126	169	84	134	162	125	108	120	73	71	0	82	87	130	123	152	164	166	153	204	172	201	141	130
s11	64	105	66	94	82	64	103	75	68	112	48	16	76	81	82	0	15	75	39	78	102	64	114	137	90	144	126	80
s12	35	71	83	134	34	82	35	117	95	63	59	56	11	92	87	15	0	14	84	121	93	83	118	139	89	136	92	53
s13	64	60	8	162	82	107	111	131	136	110	70	55	55	46	130	75	14	0	43	39	56	64	111	143	97	145	136	121
s14	116	93	2	97	70	113	102	121	90	142	97	61	76	108	123	39	84	43	0	76	19	113	55	148	88	148	110	86
s15	111	129	25	111	139	152	119	236	131	146	177	130	114	161	152	78	121	39	76	0	83	110	105	102	164	167	135	80
s16	42	82	69	55	56	91	37	91	73	92	81	54	113	134	164	102	93	56	19	83	0	9	10	75	72	109	64	36
s17	68	81	55	66	85	90	74	132	107	99	118	105	117	79	166	64	83	64	113	110	9	0	27	29	52	104	53	16
s18	53	73	130	110	57	73	80	121	77	108	83	119	75	121	153	114	118	111	55	105	10	27	0	16	29	122	89	29
s19	117	93	63	107	117	72	107	101	72	158	90	81	110	122	204	137	139	143	148	102	75	29	16	0	30	91	56	57
s20	92	117	74	11	66	68	52	85	138	104	69	111	115	87	172	90	89	97	88	164	72	52	29	30	0	27	79	90
s21	60	94	154	18	102	115	96	105	86	115	108	136	96	116	201	144	136	145	148	167	109	104	122	91	27	0	71	91
s22	52	58	110	62	37	64	105	89	41	68	59	35	88	71	141	126	92	136	110	135	64	53	89	56	79	71	0	46
s23	28	89	80	59	37	62	14	68	35	68	50	44	112	86	130	80	53	121	86	80	36	16	29	57	90	91	46	0

Scenario 8

	d0	d1	d2	d3	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
d0	0	64	72	37	65	2	4	70	12	38	53	33	30	35	108	39	33	100	115	98	74	56	96	78	44	126	8	61
d1	64	0	147	74	76	90	93	156	100	133	79	28	7	32	69	35	68	108	102	107	89	93	125	152	126	135	86	73
d2	72	147	0	84	96	92	66	116	114	154	130	92	130	96	125	58	65	90	43	72	5	37	35	77	97	101	121	56
d3	37	74	84	0	121	53	30	58	134	92	112	71	88	88	200	109	153	105	113	108	57	118	43	23	25	12	36	77
s0	65	76	96	121	0	17	33	112	17	55	75	25	48	79	103	83	58	56	100	127	94	130	105	51	106	70	30	94
s1	2	90	92	53	17	0	13	80	56	62	44	56	97	104	143	105	104	63	126	155	81	63	115	60	54	84	103	97
s2	4	93	66	30	33	13	0	64	41	35	40	15	23	19	111	43	66	116	107	103	76	84	98	127	65	131	39	56
s3	70	156	116	58	112	80	64	0	52	95	120	134	86	67	166	101	72	128	112	158	123	143	155	88	76	132	46	79
s4	12	100	114	134	17	56	41	52	0	73	68	66	72	34	169	69	82	71	79	169	82	136	100	111	135	84	46	38
s5	38	133	154	92	55	62	35	95	73	0	62	74	61	61	145	103	129	108	103	171	105	146	94	175	100	135	87	53
s6	53	79	130	112	75	44	40	120	68	62	0	42	46	70	113	72	74	108	115	149	94	77	112	92	78	85	58	113
s7	33	28	92	71	25	56	15	134	66	74	42	0	1	21	77	24	36	71	68	81	130	66	68	98	95	147	97	68
s8	30	7	130	88	48	97	23	86	72	61	46	1	0	37	107	59	22	108	55	127	88	84	89	97	65	92	49	55
s9	35	32	96	88	79	104	19	67	34	61	70	21	37	0	85	22	30	66	52	127	74	86	79	113	72	131	73	41
s10	108	69	125	200	103	143	111	166	169	145	113	77	107	85	0	115	84	152	142	154	181	140	150	162	185	169	113	113
s11	39	35	58	109	83	105	43	101	69	103	72	24	59	22	115	0	3	49	33	93	76	90	146	130	116	178	129	94
s12	33	68	65	153	58	104	66	72	82	129	74	36	22	30	84	3	0	47	85	125	53	93	70	89	117	156	89	52
s13	100	108	90	105	56	63	116	128	71	108	108	71	108	66	152	49	47	0	25	57	55	81	106	143	113	123	105	116
s14	115	102	43	113	100	126	107	112	79	103	115	68	55	52	142	33	85	25	0	60	55	50	138	97	94	146	92	85
s15	98	107	72	108	127	155	103	158	169	171	149	81	127	127	154	93	125	57	60	0	63	82	68	70	150	183	164	71
s16	74	89	5	57	94	81	76	123	82	105	94	130	88	74	181	76	53	55	55	63	0	5	38	38	46	119	110	59
s17	56	93	37	118	130	63	84	143	136	146	77	66	84	86	140	90	93	81	50	82	5	0	67	87	49	124	66	27
s18	96	125	35	43	105	115	98	155	100	94	112	68	89	79	150	146	70	106	138	68	38	67	0	12	50	113	121	33
s19	78	152	77	23	51	60	127	88	111	175	92	98	97	113	162	130	89	143	97	70	38	87	12	0	15	54	61	64
s20	44	126	97	25	106	54	65	76	135	100	78	95	65	72	185	116	117	113	94	150	46	49	50	15	0	62	34	76
s21	126	135	101	12	70	84	131	132	84	135	85	147	92	131	169	178	156	123	146	183	119	124	113	54	62	0	60	97
s22	8	86	121	36	30	103	39	46	46	87	58	97	49	73	113	129	89	105	92	164	110	66	121	61	34	60	0	46
s23	61	73	56	77	94	97	56	79	38	53	113	68	55	41	113	94	52	116	85	71	59	27	33	64	76	97	46	0

s5	57	65	153	95	38	94	61	110	54	0	78	94	139	79	119	96	153	87	132	173	149	104	126	159	116	99	105	131
s6	79	72	137	104	37	55	35	79	71	78	0	20	48	62	154	82	70	114	138	161	113	82	117	120	67	120	59	44
s7	89	2	97	106	27	65	8	52	84	94	20	0	24	50	77	86	7	25	105	99	73	121	112	95	83	113	56	25
s8	37	4	67	89	27	36	20	140	43	139	48	24	0	24	105	21	23	89	91	145	97	84	67	156	73	120	46	43
s9	33	56	147	111	25	59	41	70	33	79	62	50	24	0	51	17	66	68	119	100	70	108	87	92	67	173	99	50
s10	93	66	168	172	140	99	90	200	103	119	154	77	105	51	0	104	118	143	159	229	135	229	135	155	137	165	124	108
s11	99	34	42	162	34	87	39	113	81	96	82	86	21	17	104	0	23	39	79	85	124	93	118	143	109	103	106	41
s12	60	96	39	144	61	61	83	73	93	153	70	7	23	66	118	23	0	7	110	88	108	96	142	101	134	104	84	35
s13	68	103	26	145	95	108	88	118	140	87	114	25	89	68	143	39	7	0	5	45	29	107	81	110	126	156	101	57
s14	113	84	33	144	64	84	94	148	97	132	138	105	91	119	159	79	110	5	0	58	24	40	57	114	127	163	104	62
s15	114	116	13	113	130	122	132	182	130	173	161	99	145	100	229	85	88	45	58	0	38	86	73	82	106	126	170	77
s16	70	89	28	124	70	60	108	104	71	149	113	73	97	70	135	124	108	29	24	38	0	22	29	29	120	86	65	12
s17	106	78	81	105	76	102	46	122	107	104	82	121	84	108	229	93	96	107	40	86	22	0	58	24	121	104	93	27
s18	80	73	112	65	55	68	106	170	151	126	117	112	67	87	135	118	142	81	57	73	29	58	0	14	65	65	72	79
s19	56	172	81	25	108	123	54	94	108	159	120	95	156	92	155	143	101	110	114	82	29	24	14	0	64	41	68	110
s20	26	72	66	14	66	51	49	84	96	116	67	83	73	67	137	109	134	126	127	106	120	121	65	64	0	21	111	81
s21	121	172	182	58	78	76	63	121	123	99	120	113	120	173	165	103	104	156	163	126	86	104	65	41	21	0	71	133
s22	39	80	80	33	29	41	49	54	46	105	59	56	46	99	124	106	84	101	104	170	65	93	72	68	111	71	0	108
s23	101	76	40	54	83	76	44	144	83	131	44	25	43	50	108	41	35	57	62	77	12	27	79	110	81	133	108	0

For all the above scenarios, the cost in minutes for inspecting the substations was kept constant :

s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
60	45	30	45	30	45	30	30	45	45	45	30	45	45	45	30	45	45	30	45	45	30	30	45