Please refer to the following:
Kota László, Jármai Károly
Mathematical modeling of multiple tour multiple traveling salesman problem using evolutionary programming
APPLIED MATHEMATICAL MODELLING 39:(12) pp. 3410-3433. (2015)

MATHEMATICAL MODELING OF MULTIPLE TOUR MULTIPLE TRAVELING SALESMAN PROBLEM WITH EVOLUTIONARY PROGRAMMING

KOTA, L., JARMAI K.

Abstract: *This study describes a single phase algorithm for the fixed destination multi-depot multiple traveling salesman problem with multiple tours (mmTSP). This problem widely appears in the field of logistics mostly in connection with maintenance networks. In the first part we show the general model of the technical inspection and maintenance systems, where this problem usually emerges. We propose a mathematical model of the system's object expert assignment with the constraints like experts minimum and maximum capacity, constraints on experts' maximum and daily tours. In the second part we describe the developed evolutionary programming algorithm which solves the assignment, regarding the constraints introducing penalty functions in the algorithm. In the last part of the paper the convergence of the algorithm and the run times are presented.*

Key words: *evolutionary programming, heuristics, logistics, maintenance networks*

Laszlo Kota
University of Miskolc, Department of Material Handling and Logistics, Miskolc-Egyetemvaros, H-3515, Miskolc, Hungary, altkota@uni-miskolc.hu

Prof. Dr. Karoly Jarmai
University of Miskolc, Department of Material Handling and Logistics, Miskolc-Egyetemvaros, H-3515, Miskolc, Hungary, altjar@uni-miskolc.hu

1. Introduction

Nowadays in the field of globalized production and service industry the significance of the tightly integrated logistic systems are increasing. While in the beginning mostly the production industry gets globalized, nowadays there are multinational companies which offer even worldwide service solutions.

In the service industry the technical inspection and maintenance systems has a great importance, because they provide safety and reliable operation of production and service facilities. The most significant facilities are the communal services, water supply, electricity, district heating, fuel supply, telecommunication services or even elevators found in residential areas in large numbers.

The reliable, accident free, and economic operation require periodic technical inspections and maintenances. In these systems the inspection generally require specialized knowledge, sometimes it even requires special certificate. At elevators, which inspection and maintenance are very important from the aspect of life protection, there are governmental regulations available.

There are devices which requires periodic inspection and maintenance, for example the safety and control devices of the electricity, gas, heat, water supply networks, monitoring devices, critical network control devices which require on site supervision and maintenance.

In these networks the following tasks emerges:

- an maintainer person (called as expert hereafter) have to go the site several times in a year and to do the inspection and/or maintenance duties there,
- the maintenance and inspection tasks requires different tools and parts which has to transport to the site and/or back to the warehouse on time,
- the experts have to reside near the objects to reach lesser time expenditure and cost,
- the required materials stored in several warehouses scattered in the system,
- when an onsite non-repairable part emerge, the part has to transfer to a repair/refurbishment facility

The main problem in these type of systems is to assign the experts to the objects what they inspect and control them on everyday route while the expert have to inspect the objects and have to return to his home location at the end of the day beside with optimal number of expert to reduce the costs. This paper gives or tries to give a solution – even if some details not mentioned here due to the lack of space - a heuristic optimization of this problem, even usable in large scale systems.

2. System model of the network like inspection and maintenance systems

The network like technical inspection and maintenance systems can extend a city, a region, a country, continent wide, or even worldwide. The duties of these systems are a regular supervision of the objects on a given time period and maintenance and/or repair the parts of the objects. The effective realizations of the maintenance tasks is ensure by one or more scattered raw material and tool warehouses and repair facilities.

The role of the logistic system is to ensure the availability of the resources - experts, raw materials, tools- required by the technical inspection and maintenance tasks

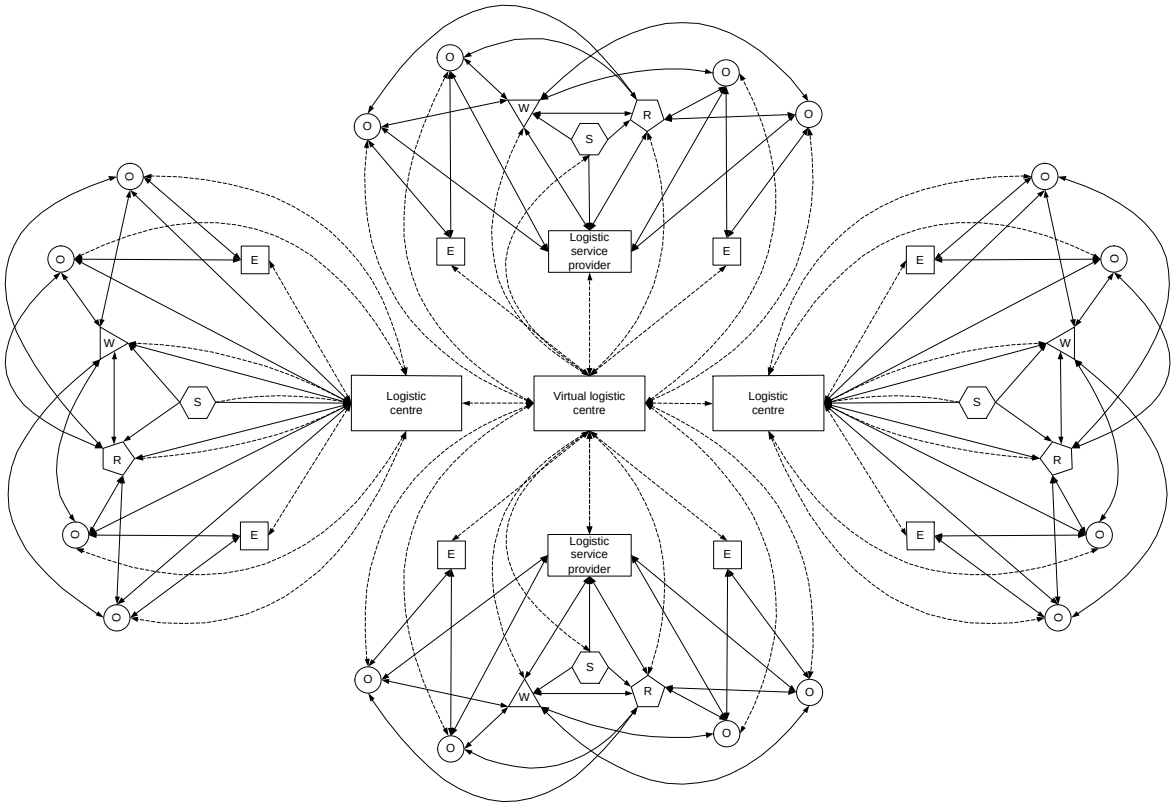


Fig. 1. General structure of a technical inspection and maintenance system
(Legend: dashed lines: Information flow, normal line: material flow, E: expert, S: supplier, O: object, R: repair facility, W: warehouse)

Regarding the geographical scatter of the required resources - experts, materials, tools - and demands - technical inspection and maintenance -, the optimal operation of the system have to ensure by a virtual logistic center or a company with a logistics center where all the material and information flow centered and where the whole system controlled.

The system is controlled by a virtual logistic center (Fig. 1.), but in smaller scale – regional or country wide systems – the core of the system, the controller facility could be a logistic center where the information processing and the material flow is simultaneously present. The virtual logistic center is in an information link with all of the system components, namely the:

– experts,
– warehouses,
– repair facilities,
– suppliers,
– logistic service providers, which provide the handling of transfer tasks in the system.

The virtual logistic center is not involved in the physical material flow, it only controls it and allocates the resources. The tasks of the virtual logistic center:

– decides which expert is chosen to the system, chooses its location,
– assigns the experts to the objects,
– schedule and register the required technical inspections and maintenances,
– chooses the location of the warehouses, repair facilities,
– chooses the suppliers,
– disposes and schedule transports to the warehouses and repair facilities,
– allocates the vehicles for transportations.

The virtual logistic center which controls the system uses complex mathematical models and optimization processes, where it minds the operational requirements, governmental regulations and many other conditions as constrains.

3. Mathematical model of the technical inspection and maintenance systems

In this article only the part of the complex model of maintenance system is shown, because the optimization covers only the object – expert assignment.
The system main parameter is the path matrix *L*, which shows the distances between the system elements. In our case the path matrix is a integrated matrix, built up from several sub-matrixes, the sub-matrices defined by the number of elements in the system.

$$L = [l_{ij}], \tag{1}$$

The assignment matrix *Y* is one of the main output parameter of the model. The assignment matrix:

$$Y = [y_{ij}], \tag{2}$$

where

– $y = \begin{cases} 1 \\ 0 \end{cases}$ according to the system elements are assigned together (1) or not (0),

Defining the y_{ij} is the assignment task which have to be solved in this complex system.

3.1 Objects

The main parameters of the objects are:

- p: is the number of the objects, it is constant in this model,
- L matrix defines the location of the objects, and the distance from the other system elements,
- $\kappa_{i\ (i=1..p)}$ is the mandatory inspection number per object,

The number of the technical inspections and maintenances could be prescribed by the maintenance plan or even law or governmental regulations in some cases where human life is endangered, like at elevators. The maintenances can't happen in an arbitrary period, there is a time period which has to be defined to every object when the next maintenance task could perform.

$$\tau^m = [\tau_i^m]_{i=1..p}. \tag{3}$$

The interval of the inspections fulfill the constraint

$$\tau_i^m * (\varepsilon_i \ - 1) \leq \vartheta, \tag{4}$$

where:

- ε_i *:* is the number of the maintenance tasks of object *i,*
- ϑ: is the examination period.

In real life of these systems the inspection and maintenance tasks are performed usually by the same expert so the special knowledge collected at the previous inspections is well utilized, so the maintenance times could be shortened.

3.1 Experts

The parameters for the mathematical description of the experts are the following:

- s: is the number of the experts, constant in most cases and in this model, but it could change for example at the case of:
  - expert leaving the job,
  - expert employment in case of system expansion,
- expert's number decreasing in case of system reduction it could be dynamic.,

The time required to travel between object *i* and *j*:

$$\tau_{i,j} = \frac{l_{i,j}}{\bar{v}}\ ; \quad \begin{matrix} i = 1..p \\ j = 1..p' \end{matrix} \tag{5}$$

where:

- $l_{i,j}$: is the distance between the object *i* and *j*,
- p : is the number of the objects,

- $\bar{v}$: the average speed of the expert.
- P: is the performance of the experts, it show how much maintenance task is performed by the expert.

Constraints:

The performance of the expert has to be between the defined minimum and maximum values:

$$P_{i\,min} < P_i \; < P_{i\,max}, \tag{6}$$

where:

$$P_i \; = \sum_{j=1}^{p} \left(Y_{12\,i,j} * \varepsilon_j \; \right). \tag{7}$$

The cycle time (τ_{max}) - generally one day – is also a constraint, in one cycle the expert visit the objects do the inspection and return to his base location:

$$\tau^t = \tau_{0,1}^f + \tau_1^k + \sum_{i=2}^{c^t} \left(\tau_i^k + \tau_{i-1,i} \right) + \tau_{q,0}^f < \tau_{max}, \tag{8}$$

where:

- τ^t: is the interval when the expert start from his base location, visits the objects and return, it is generally one day at the regional or countrywide maintenance systems and:

$$\sum_{i=1}^{T} \tau_i^t = \vartheta, \tag{9}$$

where:

- T: is the number of cycles in the ϑ interval,
- τ_{max} time interval of a cycle,
- c^t : the number of objects has to visit in the cycle *t*,
- $\tau_{0,1}^f$: the travel time to the first object from the start location,
- $\tau_{q,0}^f$: the travel time from the last object (q) to the experts base location,
- τ_i^k: the average inspection time of the object i.

The set of objects can be defined which have to inspect by the expert c:

$$O_c \; \coloneqq \left\{ o_i \; | \, Y_{12\,s,i} = 1; i = 1..p \right\}, \tag{10}$$

$$\left|O_c\right| = P_c \ , \tag{11}$$

and the subsets, the objects have to be inspected in one cycle:

$$O_c^t \subseteq O_s \ , \tag{12}$$

where:

- O_s : is an ordered set, the objects assigned to the given expert, the ordering function is:

$$o_p \in O_i; o_q \in O_i; o_p < o_q \; where \; t_{o_p} < t_{o_q}, \tag{13}$$

where:

- t_{o_p} is the inspection time of o_p,
- t_{o_q} is the inspection time of o_q,

so the set is ordered by the visiting time.

$$|O_c^t| = c_c^t \leq P_c, \tag{14}$$

$$\bigcup_{t=1}^{T} O_c^t = O_s \ , \tag{15}$$

and

$$\bigcup_{s=1}^{p} O_s^t = O \ . \tag{16}$$

However the expert performs more than one inspection on an object so the object is counted in the sets defined at (12) as many times as the number of inspection has to be performed (Fig.2, Fig.3).

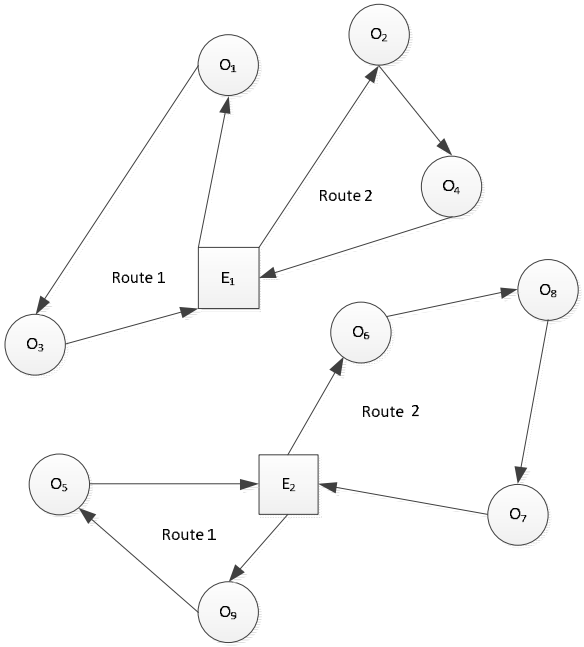


Fig. 2. A simple example of multiple routes with only one inspection at any object

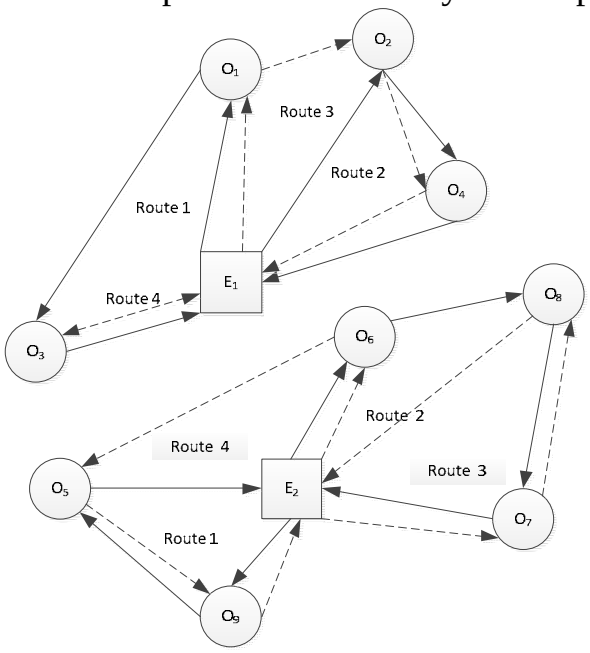


Fig. 3. A simple example of multiple routes with two inspections at the objects

To determine the interval of the inspections the following distance functions can be applied:

$$d\left(o_i; o_j \middle| o_i \in O_p^t; o_j \in O_q^t\right) = p - q, \tag{17}$$

so based on the constraint in eq. (4):

$$min\left\{d\left(o_i; o_j \middle| o_i \in O_p^t; o_j \in O_q^t\right)\right\} \geq \tau_i^m. \tag{18}$$

So the path travelled by the expert *i* in a cycle *t* can be describe as:

$$l_i^t = l_{0,O_i^t(1)} + \sum_{c=1}^{|o_i^t|-1} \left(l_{O_i^t(c),O_i^t(c+1)} \right) + l_{O_i^t(|o_i^t|),0}, \tag{19}$$

and the total path travelled by the expert *i* can be described as:

$$l_i^T = \sum_{t=1}^{T} \left[l_{0,O_i^t(1)} + \sum_{c=1}^{|o_i^t|-1} \left(l_{O_i^t(c),O_i^t(c+1)} \right) + l_{O_i^t(|o_i^t|),0} \right] = \sum_{t=1}^{T} l_p^t. \tag{20}$$

The expenditures (*C*) of the experts (*S*) in a given period (*T*) can described as:

$$C^S = \left[\sum_{j=1}^{s} \left(\sum_{t=1}^{T} l_j^t \right) \right] * c_u + \left[\sum_{j=1}^{s} P_j \right] * c_v \tag{21}$$

where:

- c_u: is the specific cost for one kilometer,
- c_v: the specific cost for an object.

Further in the article the specific cost is calculated with the multiplier 1, so only the distance is considered.
The target of the optimization is:

$$C^S \rightarrow min, \tag{22}$$

the expenditures has to be minimal.
Constraints are described in the following equations: (6), (7), (8), (9)

4 Literature on the multi-depot multiple travelling salesman problem

The problem area of the technical inspection and maintenance systems discussed here is closest to the multiple travelling salesman problems (mTSP). Within the mTSP it is the fixed destination multiple depot multiple travelling salesman problem (MDmTSP), but the solutions in the literature did not mention the additional sub-tour construction. This area is poorly researched compared to the general TSP or mTSP problems. Only the newest researches dealing with this field, and they utilize such heuristic techniques as:

- agent based modeling with probability collectives: [1] developed a multiagent model to solve the mTSP problem where they used collective memory, probability collectives, and stochastic methods, the algorithm use simple inserting, swap and elimination heuristics. They test the algorithm in two simple cases with fifteen nodes and three agents,

- genetic algorithm: [6]: They solve the problem with a two phase algorithm. In the first phase the problem was reduced from a multi-center problem into more single center problems. In the second phase the problems solved individually with a genetic algorithm. The algorithm was tested up to 99 nodes and 4 agents,
- ant colony algorithm, [5]: They used the ant colony algorithm developed by Marco Dorigo [4] and solve the general mTSP model, where artificial ants search the solution in a problem tree modeled the foraging behavior of real ants. The algorithm is compared by the solution of the Lingo 8 software and it can be seen that the ant colony heuristic algorithm is far faster and sometimes gives better solution. The algorithm was tested only up to 40 nodes and 5 agents.

The studies of optimization of complex logistics systems can be categorized into two main streams. The first stream addressed the application of meta-heuristic optimization and the second stream focuses on the application of simulation. However simulation is a useful tool to optimize systems [2], but the complexity of the system can be extremely increased the required execution time, especially in the case of global, virtual systems [3].

It can be seen that the developed methods was tested only a few nodes and they do not include the special constraints what emerge in technical inspection and maintenance system. In the field of real life logistic there are systems with over 1000 nodes or rarely but systems with over 10000 nodes exist.

5 Optimization of the complex expert object assignment of a technical inspection and maintenance system with evolutionary programming

5.1 Evolutionary programming

The problem-solving algorithms, which are a known mechanism of evolution based on evolutionary algorithms are called. The most known algorithms are the:

- genetic algorithm,
- evolutionary programming,
- evolution strategic,
- neuro-evolution.

All these algorithms have a common part: they handle a population. The population consists of individuals. One individual is one possible solution of the problem. The target is to get the best solution to the given problem. But in most of the problems the algorithms didn’t have a chance the find the optimal solution and one has to be satisfied with a quasi-optimal or a “good enough” solution.

The evolutionary programming is mainly used on heavily constrained problems. This method is also handling a population but there are no limitations on the problem representation like at the genetic algorithm where bit vectors describe the individuals. Here the problem described as the problem allows, or is it the best for computer algorithm. The pseudo code of the evolutionary programming is the following:

1. generate the first population, in most cases it is random generated,
2. calculate the population fitness values,
3. while not done
   3.1. copy the population into a temporary population,
   3.2. run the mutation operators on the temporary population,
   3.3. select the survivors for the next population,
4. end while.

In the computer solution first initialize the data, random generator, etc. Then initialize the first population. In heavily constrained problems there are two cases:

- the randomly generated population individual is invalid: it violates the constraints,
- the individual is in the feasible region: but this is a very rare case.

There are several methods to get valid individual from simply dispose invalid individuals to create special operators which retain the individual's integrity. But the simplest solution is using penalty function. In the penalty function one can regulate the algorithm which solutions are preferred.

After the creation of the initial population it has to be copied into a temporary population then the mutation operators run on the temporary population. In most cases the high impact mutations have less chance to run and the low impact operators have a bigger chance. After the mutation we have to compute the mutated individuals' fitness value and then choose the survivor individuals to the descendant population which happens with a tournament. One simple way to perform the torunament is choose two random individuals one from the original and one from the mutated population and that will survive which has less (or bigger if the fitness not normalized) fitness value, we have to repeat this until the new population not filled.

At the evolutionary programming in majority of the cases there is no crossover. In some solutions there is no meaning of this, and mostly it creates invalid individuals. So in the biological point of view this is not an evolution of one species but the evolution of many so called "population" is not valid but for the integrity of evolutionary method this naming convention is common.

5.2 The problem representation in the proposed evolutionary programming algorithm

The developed algorithm solves the fixed destination multiple depot multiple route multiple travelling salesman problem and optimize the number of salesman in one phase and can be used for large or very large problems. As there are multiple salesmen: the experts, multiple depot: all the experts have different locations, fixed destination: all the expert start and return to their initial location, and all the experts do the travel (generally) in one day cycles.

The developed solution method based on a multi chromosome technique which is not widely used in genetic algorithm but it could simply implement in the evolutionary programming.

The data structure of the optimization is built as describe in Chapter 4.1. The biggest container is the population which consists of defined constant number of individuals, which is an input parameter of the optimization.

First the algorithm initializes the optimization constants, like penalty constants, maximum tour length, maximum number of cycles, expert performance values and number of the individuals in a population, load the data files of the experts, that's define the number of the experts and their location, the objects which define the number of the objects and its location, and initialize maintenances data. Then it creates the first population with random generated individuals. It is a top down algorithm:

- first generate individuals container class,
- then generate experts container and insert it into the individuals,
- then generate chromosome and insert it into the experts so that fill the chromosome with maintenances in a random generated order.

So every expert is a chromosome that's why the algorithm named as multi chromosome algorithm. Then it calculates the fitness values of the experts and applies the penalty functions for the experts, finally applies the global penalty functions for the individual.

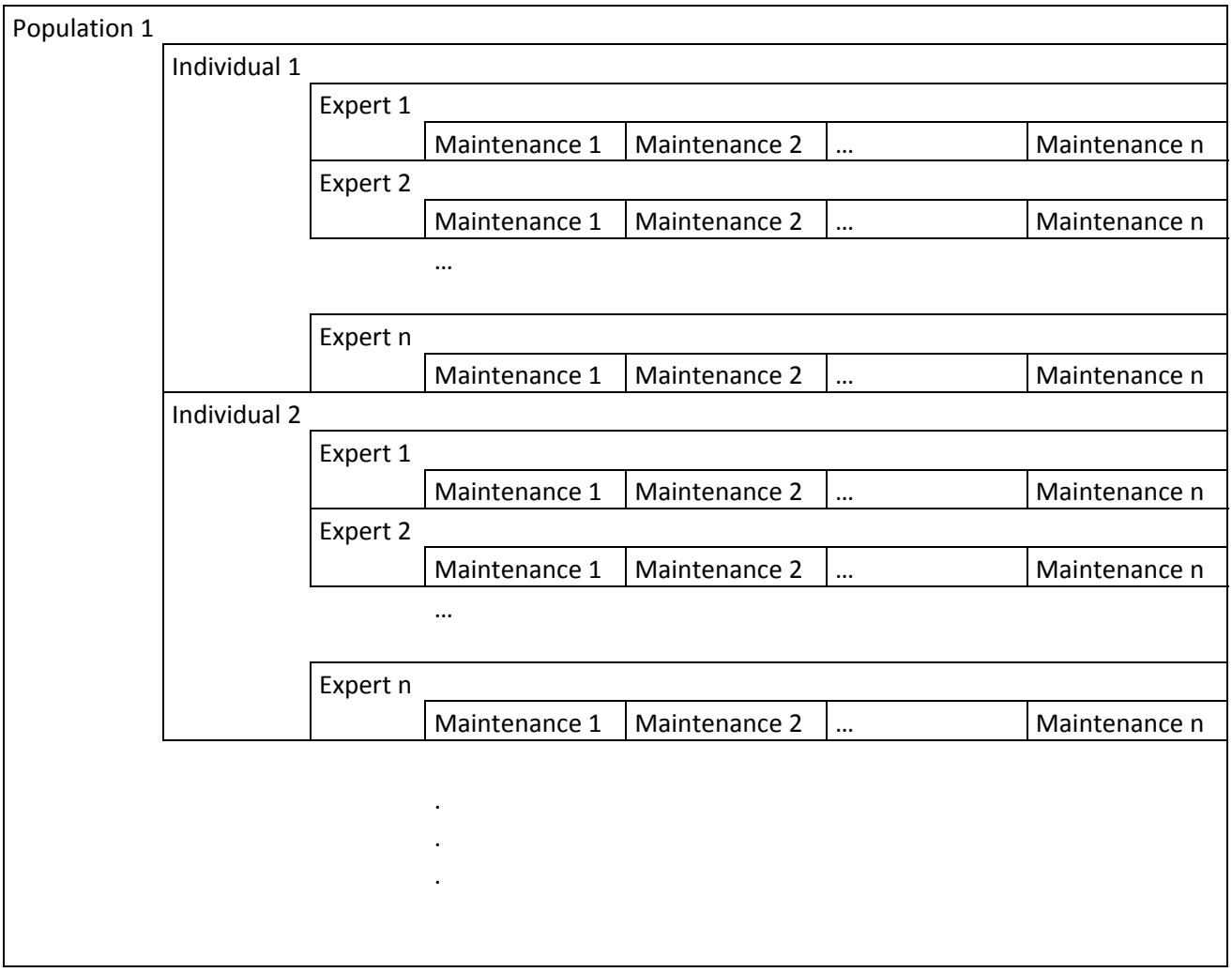


Fig. 4. The cascaded data structure of the optimization

5.1 Penalty functions

The penalty function is one of the simplest and fastest way to rate the individual, so the goodness of the actual solution. In this algorithm there are two different levels of penalty functions:

- local: the penalty function is applied to the expert,
- global: the penalty function is applied to the whole individual

penalty functions applied.

5.1.1 Local penalties

There are four different local penalty functions:

- Number of cycles penalty: when the expert do more route cycles than allowed (Eq. 5),
- Few penalty : the expert has to get a minimal number of maintenances (Eq. 8),
- More penalty: the expert cannot get more maintenances than his maximum capacity (Eq. 8),
- Near penalty: the maintenances of one object cannot be arbitrarily close to each other. (Eq. 4).

Number of cycle penalty function of the expert *i*:

$$P_C{}^I = P_{CC} * c_i \tag{23}$$

where:

- P_{CC}: constant, the penalty value of the cycle count violation
- c_i: the actual number of the expert's cycles.

Few penalty functions of the expert *i*:

$$P_F{}^I = P_{FP} * (P_{min} - P_i) \tag{24}$$

where:

- P_{FP}: constant, the penalty value of the few maintenances violation
- P_i: the actual performance of the expert *i*, the number of the maintenances

The more penalty function of the expert *i*:

$$P_M{}^I = P_{MP} * (P_i - P_{max}) \tag{25}$$

where:

- P_{MP}: constant, the penalty value of the more maintenances violation
- P_i: the actual performance of the expert *i*, the number of the maintenances

The near penalty function is the following:

$$if\ [M_i^x] - [M_{i+1}^x] < \tau^m\ \ then\ c_N = c_N{+}1$$
$$P_N{}^I\ = \sum_1^s P_{NP} * c_N \tag{26}$$

where:

- M_i^x: the maintenance *i* of the object *x*,
- P_{NP}: constant, the penalty value of the near maintenances violation,
- [ ]: index of operator, show the index of the given maintenance,
- c_N: the number of the maintenance near violations.

5.1.2 Global penalties

There are two different global penalty functions, which calculated after the local penalties:

- Scatter penalty: which applied when an object's maintenances are scattered among several experts (Chapter 3.1) ,
- Number of expert penalty: The experts employment have a fixed cost in this model. Due to this penalty functions the algorithm tries to minimize the number of employed experts.

Scatter penalty function is the following:

$$if\ count(M_i^x) < required(M_i\ )\ then\ c_P = c_P + 1$$
$$P_S\ \ = \sum_1^s P_{SC} * c_P \tag{27}$$

where:

- $count(M_i^x)$: is the number of the maintenances of the object *i* at the expert *x*
- $required(M_i\ )$: the required maintenances of the object *i*
- P_{SC}: constant, the penalty value of scattered maintenances.

If the scatter penalty switched off the algorithm is not forced to assign every maintenance of one single object to one expert but is distributes between experts.

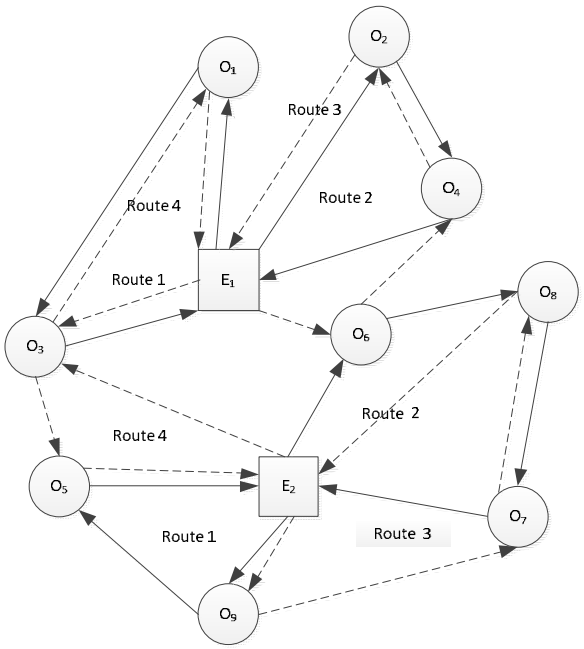


Fig. 5. An example of multiple routes with two inspections at any object without scatter penalty

Number of expert penalty function:

$$P_E = P_{EC} * \mathrm{s} \tag{28}$$

where:

- P_{EC}: constant, the cost of one experts' employment,
- s: the number of the experts.

Then the algorithm enters into the optimizations' main loop and copies the population individuals into a temporary population in order of their fitness so the best fitness value individual is copied into the first position of the temporary population. The applied operators in the algorithm are simple and more or less common to the genetic algorithm, because simpler operators results in faster algorithm.

5.2 Mutation operators

In this phase the algorithm mutates the individuals in the temporary population. There are two types of mutation operators due to the multi chromosome characteristic of the algorithm:

- inner expert mutation operators,
- cross expert mutation operators.

5.2.1 Inner expert mutation

There are three types of inner expert mutation operators:

- Gene swap: where two random genes are swapped:

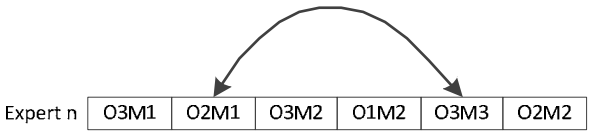


Fig. 6. Gene swap operator

– Gene sequence reversion: where the gene sequence is reversed between two random indexes

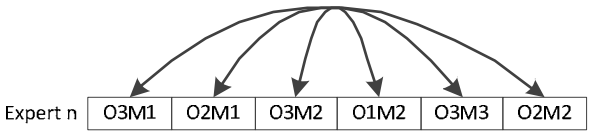


Fig. 7. Gene reversion operator

– Gene insertion operator: where a randomly chosen gene inserted into a randomly chosen position

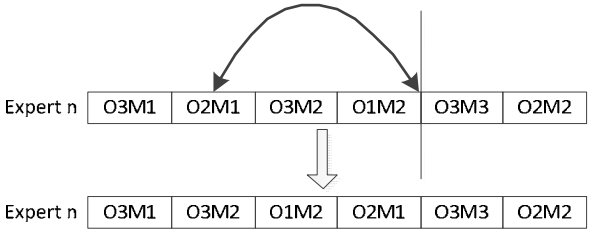


Fig. 8. Gene insertion operator

5.2.2 Cross expert mutation

Like at the inner expert mutation operators there are also three types of mutation operators exists for the mutation between experts. These are the followings:

– Cross expert gene swap: which swaps two genes between experts. The second expert and the position of the other gene are chosen randomly.

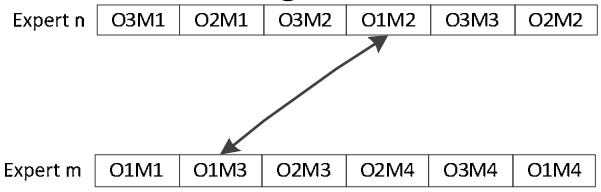


Fig. 9. Cross expert gene swap operator

– Cross expert gene sequence change: where the algorithm swapping a randomly chosen but continuous gene sequence with a randomly chosen expert also randomly chosen gene sequence:

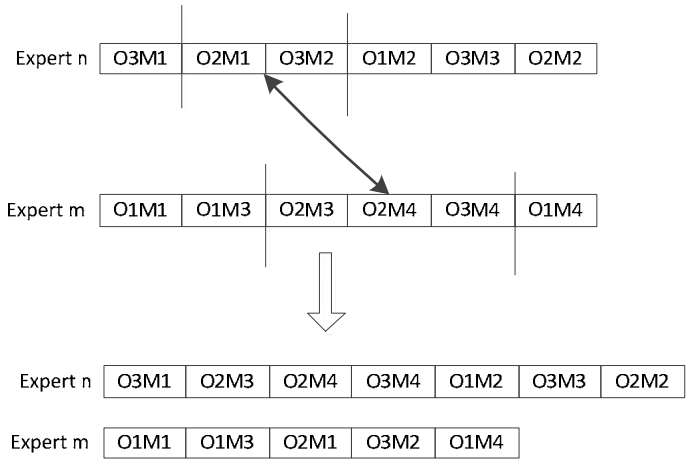


Fig. 10. Cross expert gene sequence swap operator

- Cross expert gene contraction: where random amount of genes inserted to a randomly chosen expert from the end of the chromosome. There are two types of chromosome contraction operators:
  - The first type contracts a randomly chosen chromosome (expert n) by displacing random amount genes from the end of the chromosome. It chooses a random length gene sequence from the end of the chromosome and inserts it at the end of another randomly chosen chromosome.

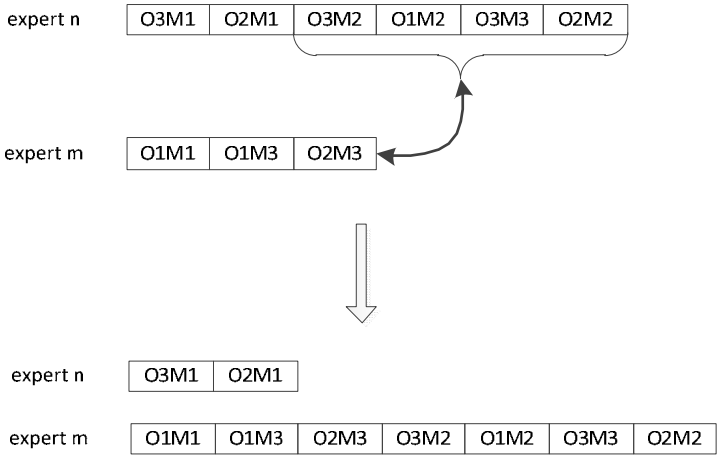


Fig. 11. Cross expert chromosome contraction Type 1

  - The second type of contraction operator displacing random amount of genes from the end of a randomly chosen chromosome like the first operator but it spread the genes between randomly chosen chromosomes one by one type

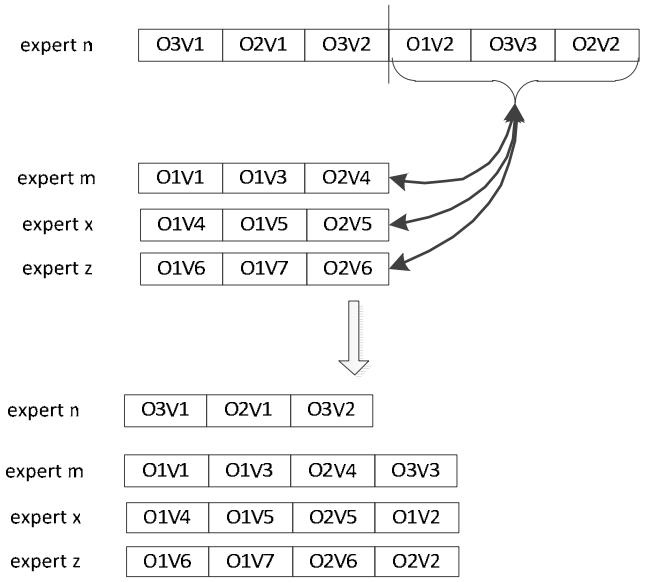


Fig. 12. Cross expert chromosome contraction Type 2

The mutations are chosen randomly for every expert with parameterized probability and the algorithm allows a probabilistic value for the no mutation also.

5.3 Survivor selection

First the algorithm searches for the best individual and copy it to the descendant population. It is called elitism; the best individual (elitist) always survives. The evolutionary programming typically uses stochastic tournament survivor selection. The simplest of the tournament selection is when randomly choose two individuals (1+1 strategy), one from the original and one from the mutated population and the fittest wins, that individual will be copied into the next generation of the individual. This process is repeated until the next population is filled.

Avoiding the local optimums all of the genetic methods try to maintain the diversity among the individuals. In this algorithm one can parameterize how much individual in the descendant population will be filled with random generated individuals

The tournament process:

- randomly choose an individual from the original and another from the mutated population,
- the individual with the best fitness value is inserted into the descendant population,
- repeat the process until the population is filled up to the required count.

The selection process fills the descendant population up to a parameterized value, the rest is filled with random generated individuals, which helps to avoid local optimum.

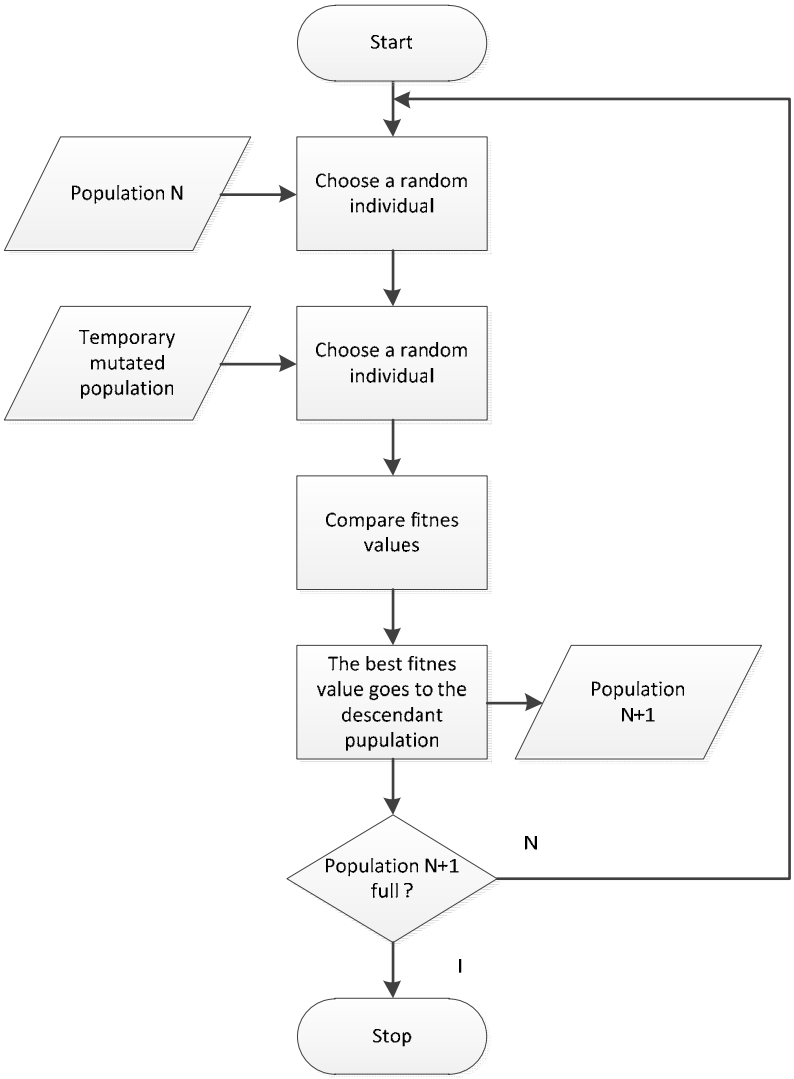


Fig. 13. Algorithm of the tournament process

5.4 Examination of the two contraction operators

All the examination in this chapter is based on the “ring” instance presented in Chapter 7.
The examination of the two contraction operators gives contradictory results. So we had analyzed the operators through several runs with the same parameters.
First examination: 3 experts 48 nodes number of examination 2-4.

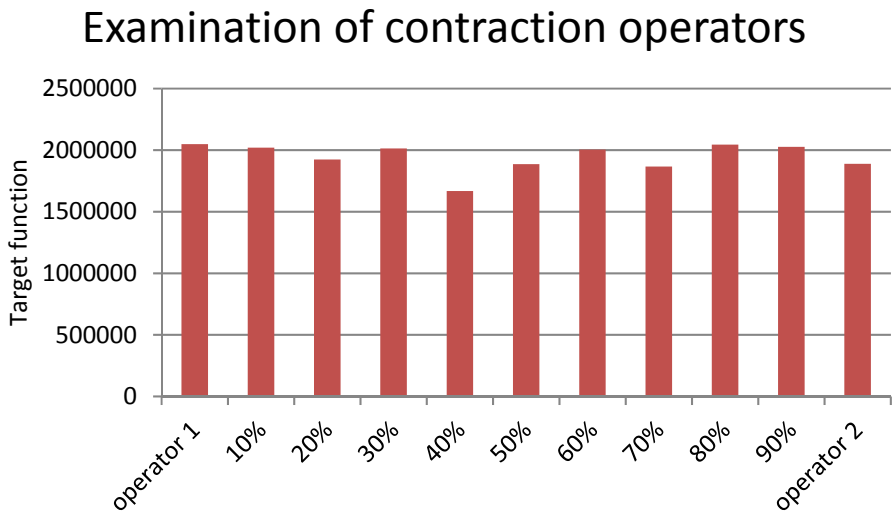


Fig. 14. First examination of contraction operators

First examination: 3 experts, 48 nodes, number of examination is 5-10 (Fig .14)

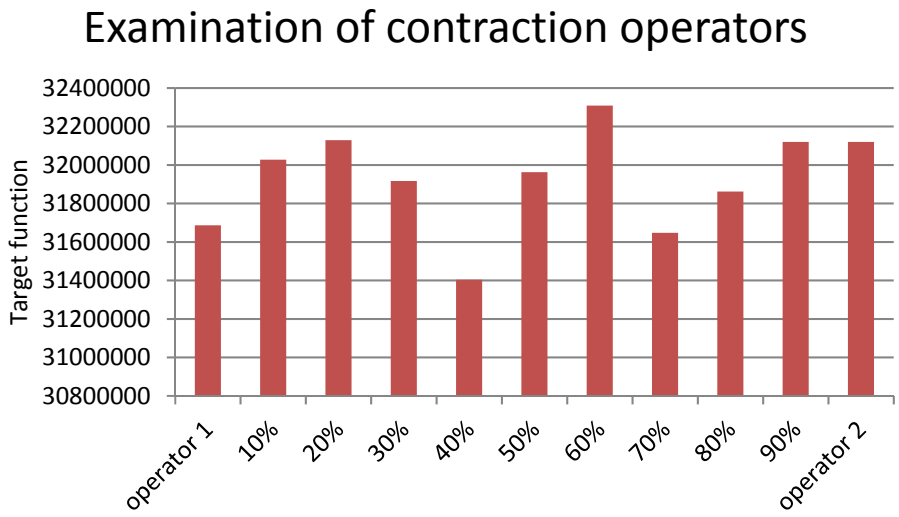


Fig. 15. Second examination of contraction operators

Second examination: 3 experts, 48 nodes, number of examination is 10-15 (Fig. 15)

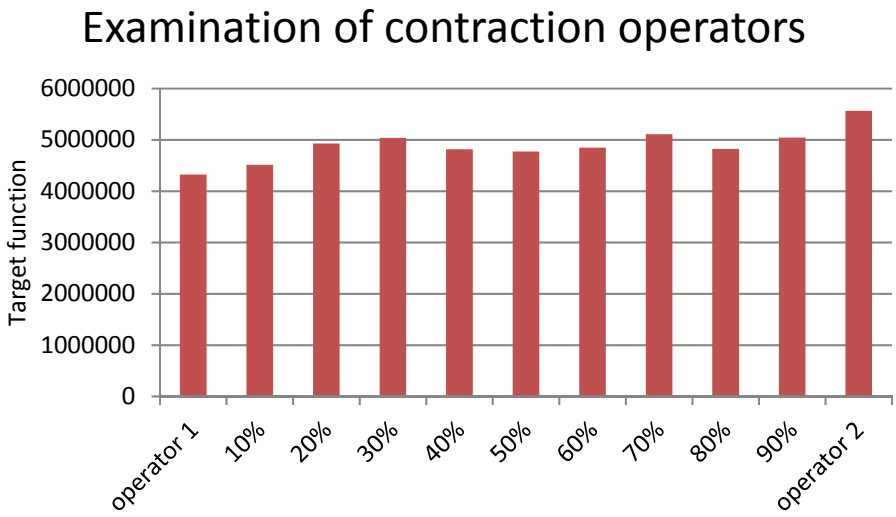


Fig. 16. Second examination of contraction operators

The examination shows that in some cases when the second contraction operator is also used with 40 percent probability gives better results. So another examination was performed with the 40 percent probability of the second operator with increasing problem size (Fig. 16.)

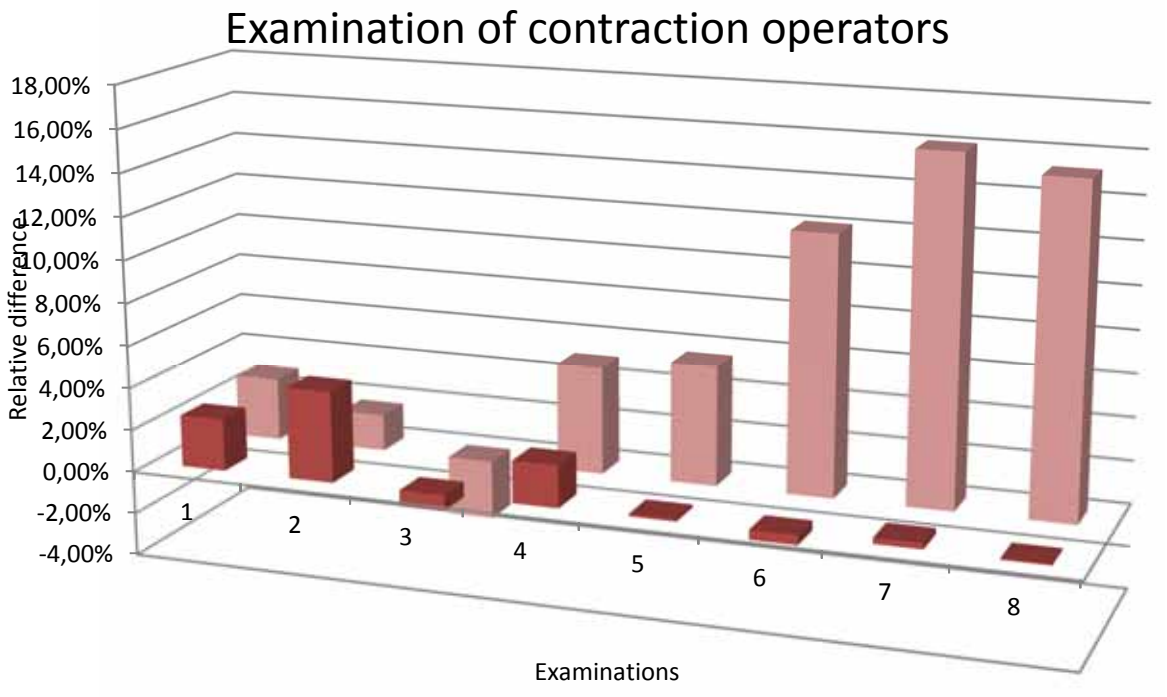


Fig. 17. Examination of contraction operators, with 40% second operator probability

The examination shows that the usage of the second operator with 40% probability is gives better result just in some random cases and the improvement only about 1% so the usage of a second contraction operator is not justified

6. Parallelization of the fitness calculation

Testing the algorithm with large scale systems it can be seen the running times grows exponentially so the parallelization of the algorithm or the parts of the algorithm is obvious to run on multiprocessor computers or even computing clouds.

The algorithm analysis shows that the mutation and the fitness calculation are the two most used and most time consuming procedures. The mutation process due to the multi-chromosome design isn't suitable for parallelization, because the global mutation operators process the entire individual so the locks of the individuals' memory area could slow the whole process, so the benefits can hardly estimate. But the fitness calculation can easily parallelize because it does not do any data modification.

The gain of the parallelization isn't obvious, because the operating system assign the threads to the available processor cores and schedule them , but the administration of the threads costs time and resources so at smaller tasks the administration cost can easily exceed the gains.

The fitness calculation can be divided into two parts. First part is the calculation of the multiple routes. This routine uses only one chromosome but it uses the path matrix which has to be copied into every computing node on a multicomputer system or it can be accessed thru shared memory in a single computer with multiple processors. In large scale systems the path matrix can be very large and it requires large memory at every computing node.

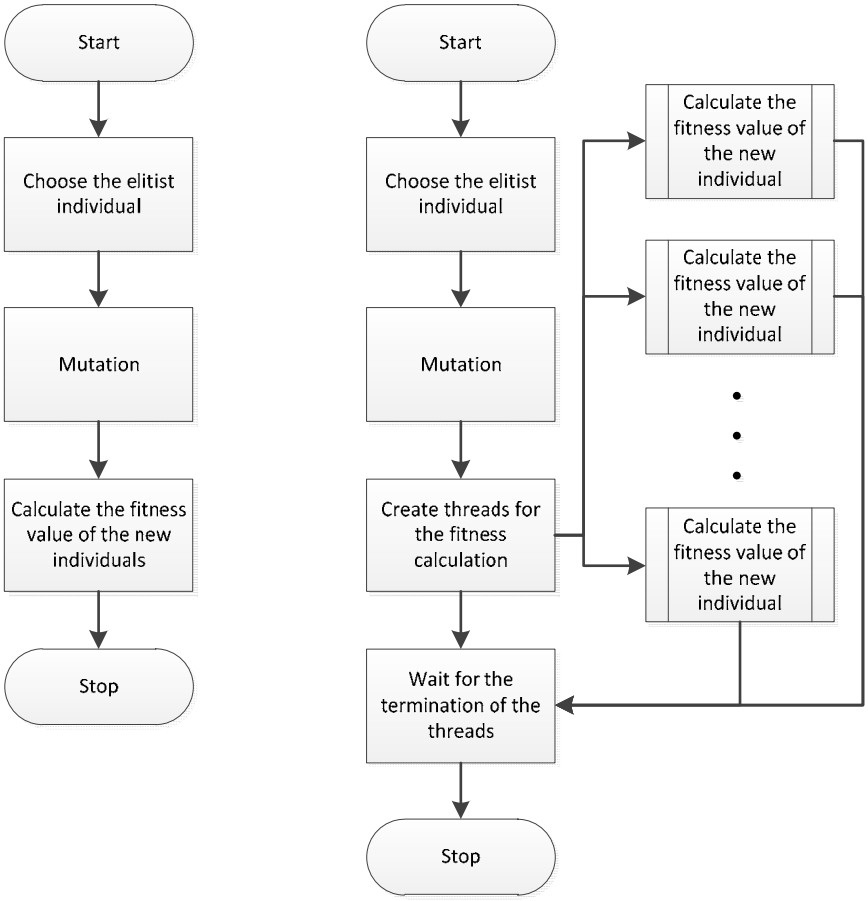


Fig. 18. Parallelization of the fitness calculation (a, serial execution, b, parallel execution)

6.1 Examination of the parallelization

The examination of the parallelization needs large scale problems so I choose the TSPLIB library for testing instances [7]. The examination methodology was:

- random generator initialized with the same value,
- same tsplib data,
- equal number of iterations.

The examination was run on a four core computer with hyperthreading technology (Table 1.).

The results:

No. of iterations	Instance (tsplib)	Number of nodes	Paralellized algorithm	Run time [mm:ss]	Speed increase [%]
100	dsj1000	1000	no	00:38	
100	dsj1000	1000	yes	01:04	-40,63
100	pr2392	2392	no	01:32	
100	pr2392	2392	yes	01:45	-12,38
100	pcb3038	3038	no	02:03	
100	pcb3038	3038	yes	02:08	-3,91

100	fl3795	3795	no	02:43	
100	fl3795	3795	yes	02:41	1,24
100	fnl4461	4461	no	03:15	
100	fnl4461	4461	yes	03:05	2,63
100	rl5934	5934	no	04:32	
100	rl5934	5934	yes	04:04	11,48
100	pla7397	7397	no	05:55	
100	pla7397	7397	yes	05:19	11,29
100	rl11849	11849	no	12:01	
100	rl11849	11849	yes	10:03	19,57
100	usa13509	13509	no	13:56	
100	usa13509	13509	yes	11:49	17,91

Table 1. Examination of parallelization with small number of individuals

The diagram (Fig 19.) clearly shows that the administration cost of the threads at small problem sizes is exceeding the obtainable gain of the parallelization process. So the parallelization on the test computer is economical above 3500 nodes with these input parameters and settings. The results also show that the gain with four processor cores is not exceed 20 percent.

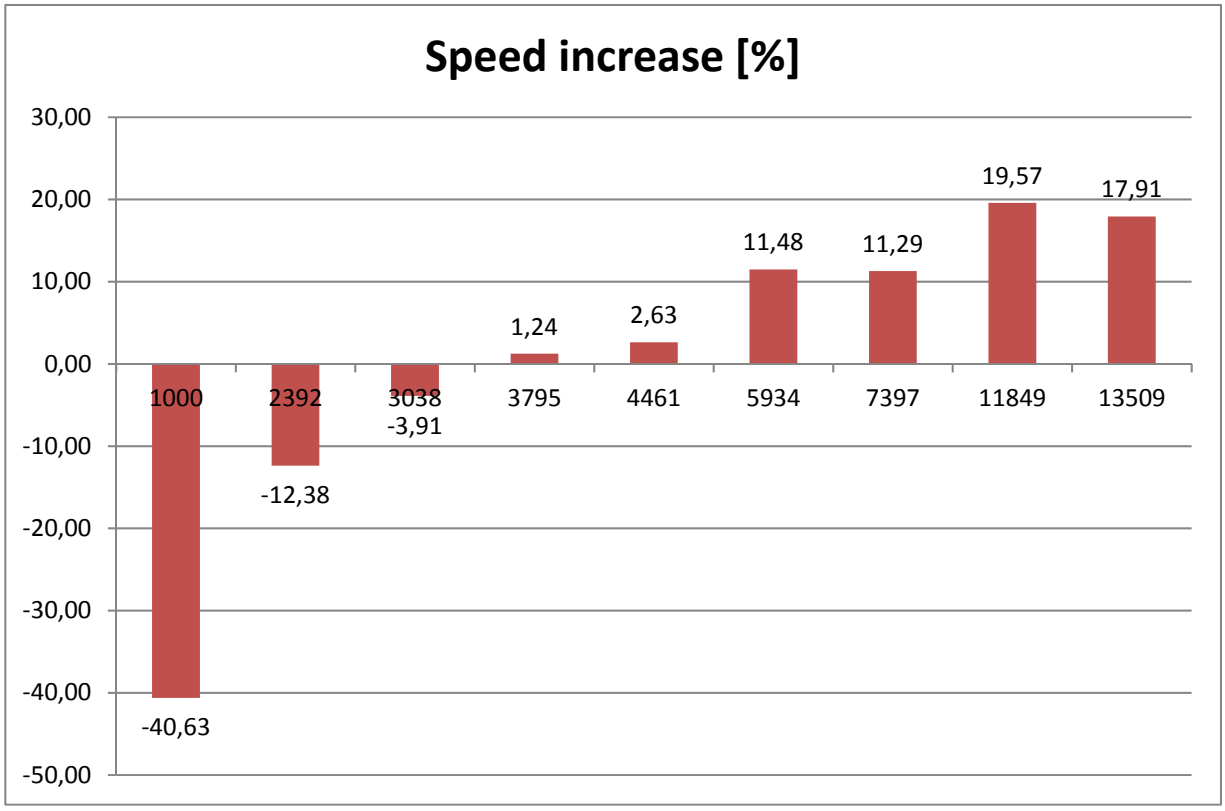


Fig 19. Speed increase in the function of number of nodes

Increasing the problem size with increasing the number of individuals up to 5 times the thread administration costs increase more (Table 2.).

No. of iterations	Instance (tsplib)	Number of nodes	Parallelized algorithm	Run time [mm:ss]	Speed increase [%]
50	dsj1000	1000	no	01:20	
50	dsj1000	1000	yes	03:32	-62,26
50	pr2392	2392	no	03:25	
50	pr2392	2392	yes	06:02	-43,37
50	pcb3038	3038	no	04:23	
50	pcb3038	3038	yes	06:55	-36,63
50	fl3795	3795	no	06:10	
50	fl3795	3795	yes	08:22	-26,29
50	fnl4461	4461	no	07:18	
50	fnl4461	4461	yes	08:29	-13,95
50	rl5934	5934	no	09:54	
50	rl5934	5934	yes	08:35	15,34
50	pla7397	7397	no	12:52	
50	pla7397	7397	yes	11:02	16,62
50	rl11849	11849	no	24:02	
50	rl11849	11849	yes	19:15	24,85
50	usa13509	13509	no	28:18	
50	usa13509	13509	yes	22:30	25,78

Table 2. Examination of parallelization with large number of individuals

The examination with increased number of individuals shows that the administration cost of the threads is increased due to the larger thread number so the speed gain value in the diagram shifted to the right. The speed gain threshold limit is doubled compared to the first test, it is about 5200 nodes. So it is obvious solving large or very large scale problems often requires multiprocessor machines or computing clouds or specialized GPU farms depending of the tasks to solve.

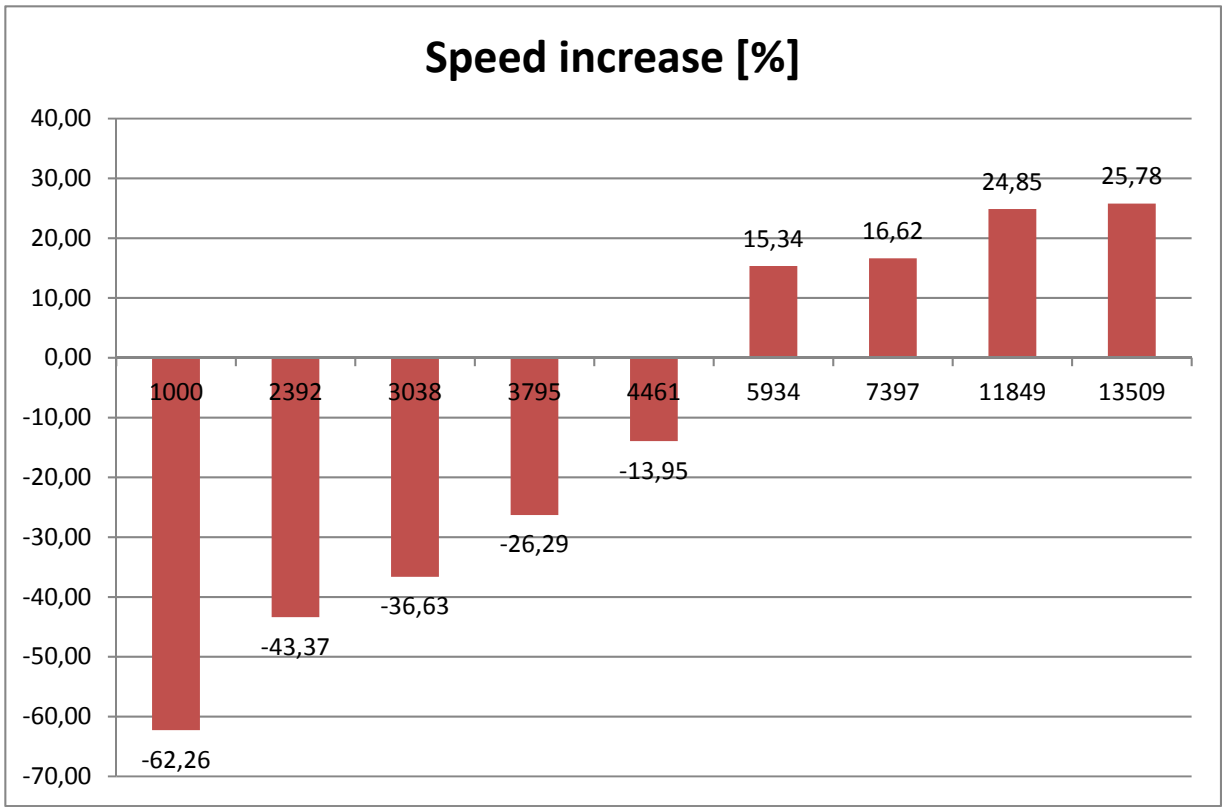


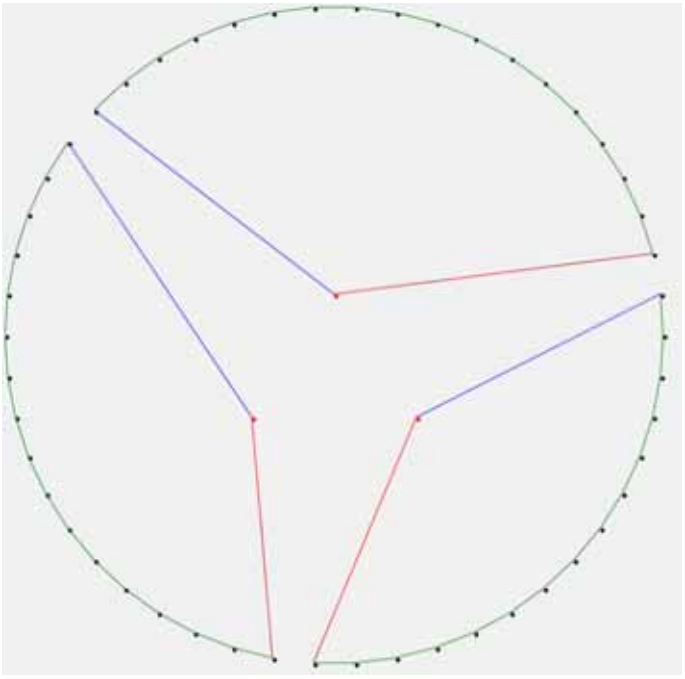
Fig 20. Speed increase in the function of node at larger problems

7 Results

The main target of the algorithm was to optimize logistic costs of large scale maintenance systems with multiple routes. So it was obvious that a computer program was needed not just test and refine the algorithm, but for further improvements and for solving real life problems also.

7.1 Test instance with three experts

This test instance uses three experts and 48 nodes, all nodes must inspect only once so the results can easily prove by naked eye.



Iteration number	35457
Run time	48 min 33 sec
Penalty	0
Cost	4484,47

Fig 21. Test instance with three experts

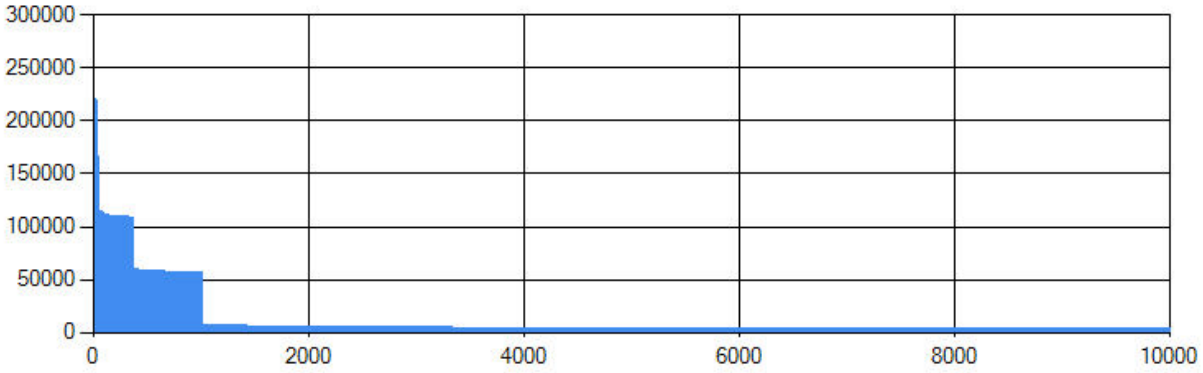
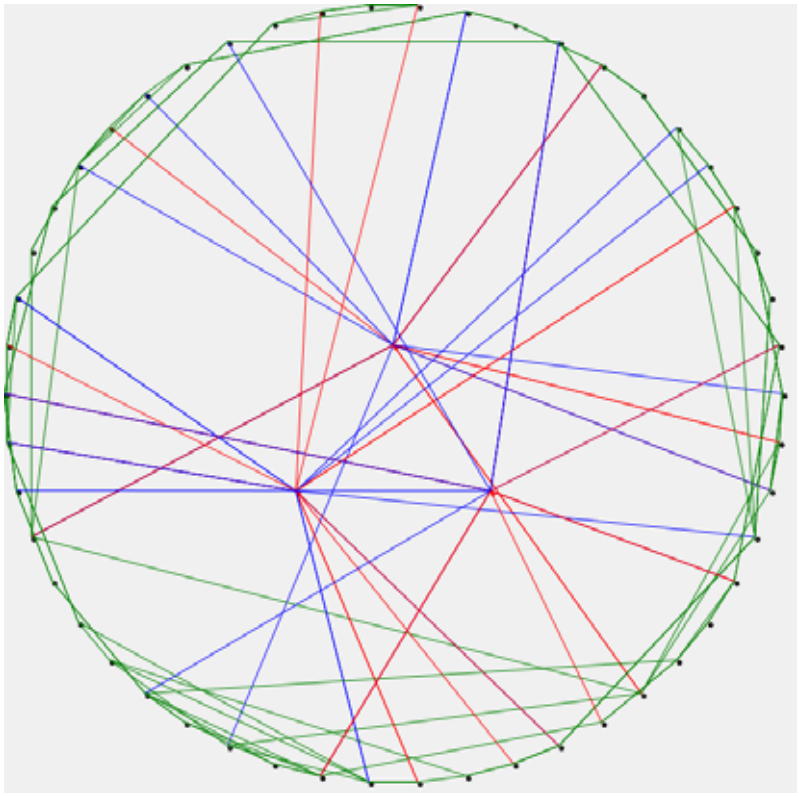


Fig 22. Convergence of the solution

7.2 Complex test instance with three experts

This test instance uses three experts and 48 nodes, but in this case all nodes must inspect 2-4 times (randomly).



Iteration number	50000
Run time	1 h 11 min 9 sec
Penalty	5
Cost	760731,64

Fig 23. Test instance with three experts

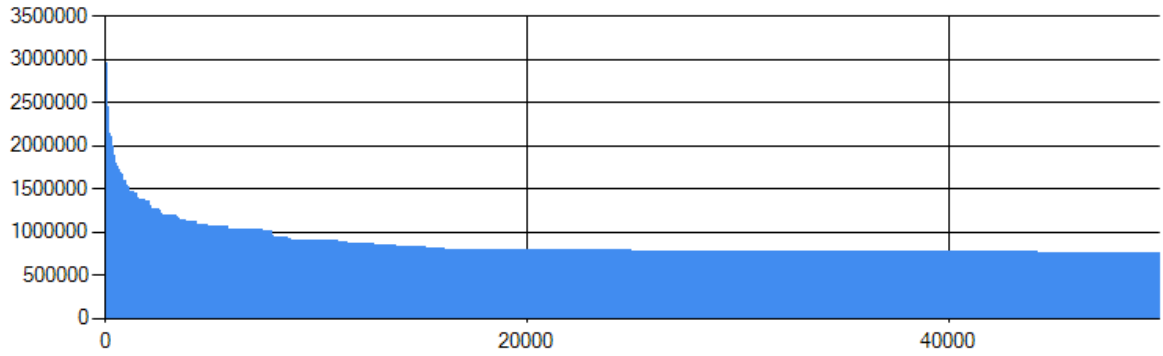


Fig 24. Convergence of the solution

8. Sensitivity analysis

8.1 Examine the population size

The first examination is the population size. The question is: how the solution changes if the population size is increased and changes the ratio of the randomly generated individuals. How the algorithm is escapes from local optimum when the ratio of the randomly generated individuals changed. The test was performed on an instance with 48 nodes, and 3 experts, 1000 iterations and with 16-18 maintenances (Table 3-4.).

The results of the sensitivity analysis, the best target function value marked with green color

		Population size						
		10	20	25	50	100	200	300
Random individuals number	0	262147,41	160300,48	158774,1	158330,53	158421,46	158033	158219,08
	5	367630,06	262188,85	261761,12	210446,86	160049,98	107682,99	158103,69
	10	-	365443,7	314261,95	211049,09	158930,41	158805,26	158249,05
	20	-	-	470334,75	263240,72	159747,81	209857,65	158833,65
	50	-	-	-	-	314076,62	262342,62	210368,71
	100	-	-	-	-	-	315404,74	313768,66
	200	-	-	-	-	-	-	366780,29
	300	-	-	-	-	-	-	-
	400	-	-	-	-	-	-	-
	500	-	-	-	-	-	-	-
	600	-	-	-	-	-	-	-
	700	-	-	-	-	-	-	-
	800	-	-	-	-	-	-	-
	900	-	-	-	-	-	-	-

Table 3. Changes of the target function in the function of the population size and the number of random individuals (part 1)

		Population size						
		400	500	600	700	800	900	1000
Random individuals number	0	157955,21	107545,32	107215,26	158309,14	157414,73	107832,6	106779,7
	5	158538,91	157945,82	108574,83	107437,69	107556,71	157603,09	56174,49
	10	158907,74	158227,07	158565,84	158039,39	107418,38	157906,86	106265,12
	20	159237,94	158423,14	158033,83	158684,29	106752,63	107745,62	157781,22
	50	159222,57	210896,47	159411,16	158886,62	158843,58	158450,84	158429,52
	100	261921,75	261951,37	211042,73	211105,01	158283,73	261510,72	158332,04
	200	366377,61	315208,69	314279,49	364707,29	211106,92	210566,73	261773,05
	300	416787,63	316873,58	313776,11	314413,83	313991,51	212120,57	261151,58
	400	-	468406,88	415795,6	365993,76	315628,49	314367,26	261779,57
	500	-	-	417931,19	417772,18	367165,7	313930,85	315149,76
	600	-	-	-	470131,99	469421,26	366624,13	366926,91
	700	-	-	-	-	469115,21	418974,4	417495,43
	800	-	-	-	-	-	521295,51	520073,15
	900	-	-	-	-	-	-	469285,56

Table 4. Changes of the target function in the function of the population size and the number of random individuals (part 2)

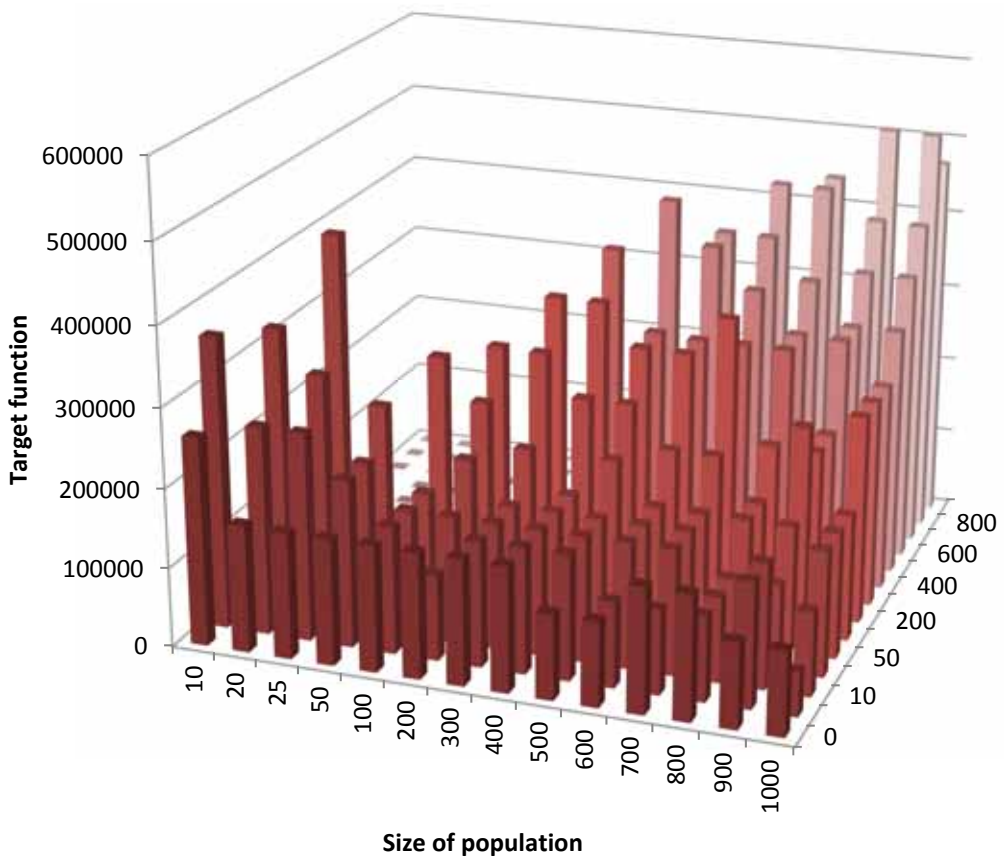


Fig 25. Changes of the target function in the function of the population size and the number of random individuals

The results show that the increases of the number of the randomly generated individuals worsen the target function because the random individuals lower the discovered search space at this small example. At greater problems the effect of the random individuals are more reasonable, at the examinations showed the number of random individuals is about 2% of the problem size.

If the problem is examined in equal size of the search space:

Iteration number	Population size	Examined number of individuals
10000	10	100000
5000	20	100000
4000	25	100000
2000	50	100000
1000	100	100000
500	200	100000
333	300	99900
250	400	100000
200	500	100000
166	600	99600
142	700	99400

125	800	100000
111	900	99900
100	1000	100000

Table 5. Calculating the number of iterations

The results are the following: (the best value marked in every column)

		Population size						
		10	20	25	50	100	200	300
Number of random individuals	0	55951,46	158334,55	158008,94	157770,09	158421,46	209782,38	210728,87
	5	210556,4	158660,78	158357,34	159260,35	160049,98	262039,25	211126,09
	10	-	210818,29	211100,73	159313,42	158930,41	160241,57	263370,5
	20	-	-	315210,41	262172,15	159747,81	262550,83	211108,47
	50	-	-	-	-	314076,62	314856,48	262600,5
	100	-	-	-	-	-	366792,2	366268,57
	200	-	-	-	-	-	-	468790,58
	300	-	-	-	-	-	-	-
	400	-	-	-	-	-	-	-
	500	-	-	-	-	-	-	-
	600	-	-	-	-	-	-	-
	700	-	-	-	-	-	-	-
	800	-	-	-	-	-	-	-
	900	-	-	-	-	-	-	-
		10000	5000	4000	2000	1000	500	333
		Number of iterations						

Table 6. Changes of the target function in the function of the population size and the number of random individuals (part 1)

		Population size						
		400	500	600	700	800	900	1000
Number of random individuals	0	212248,01	262190,95	313917,48	264179,11	314288,39	315548,66	366054,69
	5	211217,26	212174,58	263090,52	263758,78	212794,85	314842,36	314462,27
	10	212798,04	263654,66	315512,02	263160,72	264119,96	366713,93	315107,07
	20	264379,08	313330,9	263381,38	313940,42	314754,14	265100,54	315326,35
	50	314916,29	314182,9	315912,94	315163,88	263890,49	315607,6	365578,87
	100	365307,11	315883,01	314519,21	314939,88	365646,09	367097,56	314419,16
	200	419323,32	418746,37	417607,54	366973,62	366634,96	365900,83	418172,79
	300	520380,75	469494,91	417193,98	366382,58	418090,33	366859,98	470574,46
	400	-	521841,86	520747,43	520786,31	418626,89	471016,59	469912,09
	500	-	-	573818,67	470332,43	522793,76	521398,17	521406,94
	600	-	-	-	573804,97	521275,66	471914,58	571301,22
	700	-	-	-	-	572568,34	521201,2	522919,64
	800	-	-	-	-	-	522945,92	572292,19
	900	-	-	-	-	-	-	522050,56
		250	200	166	142	125	111	100
		Number of iterations						

Table 7. Changes of the target function in the function of the population size and the number of random individuals (part 2)

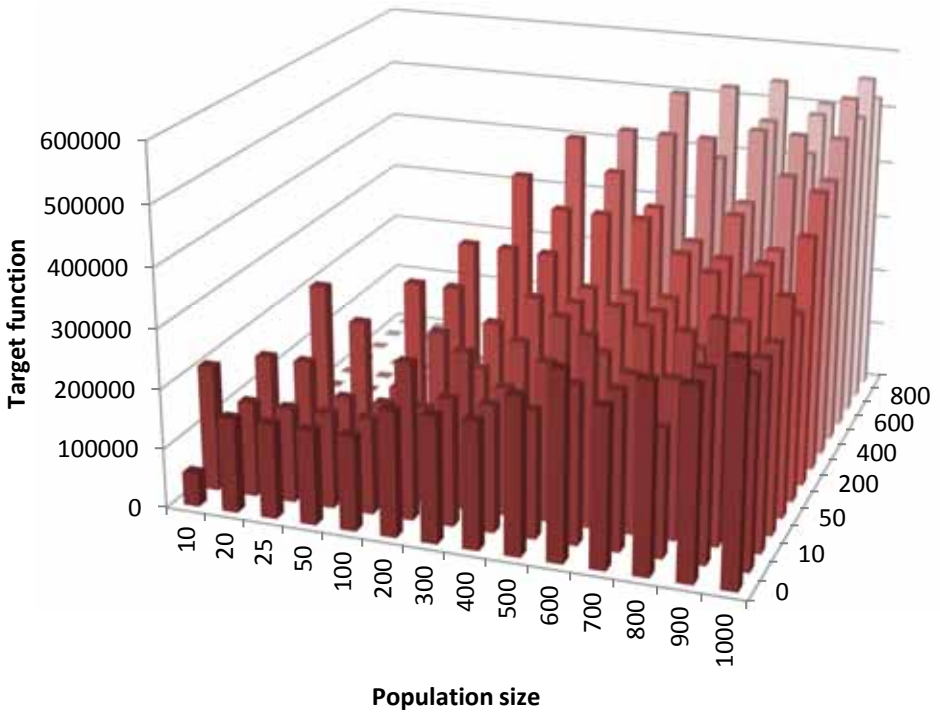


Fig 26. Changes of the target function in the function of the population size and the number of random individuals

These results (Table 6-7.) show the effect of the randomly generated individuals. The optimal ratio is in the 0-3% area. The test shows some unconventional results also but these are random perturbations. The test run shows that it is not worth to give higher the number of individuals than the number of nodes because in that case the runtime could grow considerably. And if the number of iterations are decreased the effect of the selection isn't prevail which sorts the inadequate individuals out of the population.

9 Conclusions

The algorithm described in this paper is very well applicable in the field of technical inspection and maintenance systems, because it regards the special constraints of this area like the one object one expert or one object more experts assignment constraint. However the complexity of the problem, which well described by the multitude of constraints, requires a general evolution method, which can integrate several constraint functions with relative easily. As the analysis of the test examples shows at large scale problems the usage of a computer cloud with high computation capability is highly recommended.

10 Further improvements

The optimization method has great potentials toward the further improvements. First some improvements in the method himself:

- try new mutation methods,
- try new survivor tournament selection methods.

Second some improvements which have a great impact on speed:

- massive parallelization in a computing cloud,
- examine the implementation possibilities on a fast graphics processing units (GPU) or a GPU cluster

10. Acknowledgements

The research was supported by the TÁMOP 4.2.1.B-10/2/KONV-2010-0001 entitled: Increasing the quality of higher education through the development of research - development and innovation program at the University of Miskolc.

11. References

[1.] Anand J. Kulkarni, K. Tai (2010): Probability Collectives: A multi-agent approach for solving combinatorial optimization problems, *Applied Soft Computing 10*, pp.: 759–771, doi:10.1016/j.asoc.2009.09.006

[2.] Bányai, Á. (1999) Das virtuelle Logistikzentrum als Koordinator der logistischen Aufgaben. In: *Modelling and optimization of logistic systems – Theory and practice*, Bányai, T. & Cselényi, J. (Eds.), pp. 42-50, ISBN 963 661 402 4, Published by the University Miskolc.

[3.] Bányai, T. (2009): Optimisation of U-shaped flexible manufacturing cells. In: *Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium* "Intelligent Manufacturing & Automation: Focus on Theory, Practice and Education", Katalinic, B. (Ed.), pp. 761-762, ISSN 1726-9679, ISBN 978-3-901509-70-4, Vienna, Austria, November 2009, Published by DAAAM International Vienna

[4.] Dorigo M., Stützle T. (2004): *Ant Colony Optimization*, MIT Press., ISBN 0-262-04219-3

[5.] Soheil Ghafurian, Nikbakhsh Javadian (2011): An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems, *Applied Soft Computing 11*, pp.: 1256–1262

[6.] Suk-Tae Baea, Heung Suk Hwanga, Gyu-Sung Choa and Meng-Jong Goan (2007): Integrated GA-VRP solver for multi-depot system, *Computers & Industrial Engineering* Volume 53, Issue 2, Pages 233-240, doi:10.1016/j.cie.2007.06.014

[7.] University of Heidelberg, Institut für Informatik: Library of sample instances for the TSP (and related problems) from various sources and of various types, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/