Applied Mathematical Modelling 39 (2015) 3410–3433

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Mathematical modeling of multiple tour multiple traveling salesman problem using evolutionary programming

L. Kota *, K. Jarmai

University of Miskolc, Department of Material Handling and Logistics, Miskolc-Egyetemvaros, H-3515 Miskolc, Hungary

ARTICLE INFO

Article history:
Received 8 November 2011
Received in revised form 21 October 2014
Accepted 24 November 2014
Available online 5 December 2014

Keywords:
Evolutionary programming
Heuristics
Logistics
Maintenance networks

ABSTRACT

This study describes a single phase algorithm for the fixed destination multi-depot multiple traveling salesman problem with multiple tours (mdmTSP). This problem widely appears in the field of logistics mostly in connection with maintenance networks. The general model of the technical inspection and maintenance systems is shown in the first part, where the solution of this problem is an important question. A mathematical model of the system's object expert assignment is proposed with the constraints typical of the system, like experts' capacity minimum and maximum and constraints on maximum and daily tours of the experts. In the second part, the developed evolutionary programming algorithm is described 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 and some examination of the parallelization are presented.

1. Introduction

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

The operation of these systems requires periodic technical inspections and maintenance. The inspection generally requires 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 [1].

There have been many papers published dealing with multiple traveling salesman problem like [2] where the application of Google maps introduce a novel dimension into the application, but in out paper we treated the more complicated multiple route multiple traveling salesman problem.

There are devices which require 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 [3].

The main problem in this type of systems is to assign the experts to the objects what they inspect and control the experts on everyday route. The experts have to inspect the objects and to return to his home location at the end of the day. The optimal number of expert reduces the costs. This paper presents a heuristic optimization of this problem, even usable in large scale systems.

* Corresponding author.
E-mail addresses: altkota@uni-miskolc.hu (L. Kota), altjar@uni-miskolc.hu (K. Jarmai).

http://dx.doi.org/10.1016/j.apm.2014.11.043
0307-904X/

2. Model of the network like inspection and maintenance systems

In these systems the maintenance tasks are ensured by one or more raw material and tool warehouses and repair facilities scattered over a large area.

The task of this logistic system is to provide the reliable operation and ensure the availability of resources – experts, raw materials, tools.

The optimal operation of the system has to be ensured by a virtual logistic center [4] due to the scattered resources and system elements, where the material and information flow is monitored and controlled.

These type of systems are often controlled by a virtual logistic center (Fig. 1), but in smaller scale – regional or countrywide systems – the controller facility, the core of the system, could be a conventional logistic center where not just the information processing but the material flow are also present.

The virtual logistics center controls the system using optimization processes, which requires complex mathematical models regarding constraints like operational requirements, government regulations and many other conditions.

3. Mathematical model of the technical inspection and maintenance systems

The pure multiple traveling salesman problem has a very extensive literature, but optimizing such a large systems with a lot of input parameters and boundary conditions cannot be found. In the literature, one can find researches with multiphase optimizations [5], like clustering or partitioning first and apply different multiple traveling salesman (MTSP) methods [6,7]. The clustering [8,9] and the partitioning [10] are widely used thanks to the speed of the algorithm but these methods are less suitable to found the global optimum due to the nature of the multiphase optimizations. The method presented here can optimize large systems with one phase algorithm. Above all, it handles the special constraints of the technical maintenance systems which we encountered in industrial projects.

In this article, only part of the complex model of maintenance system is shown because the optimization covers only the object – expert assignment.

In real logistic system inspection, maintenance and examination tasks are different, some require and some not require spare parts to build in.

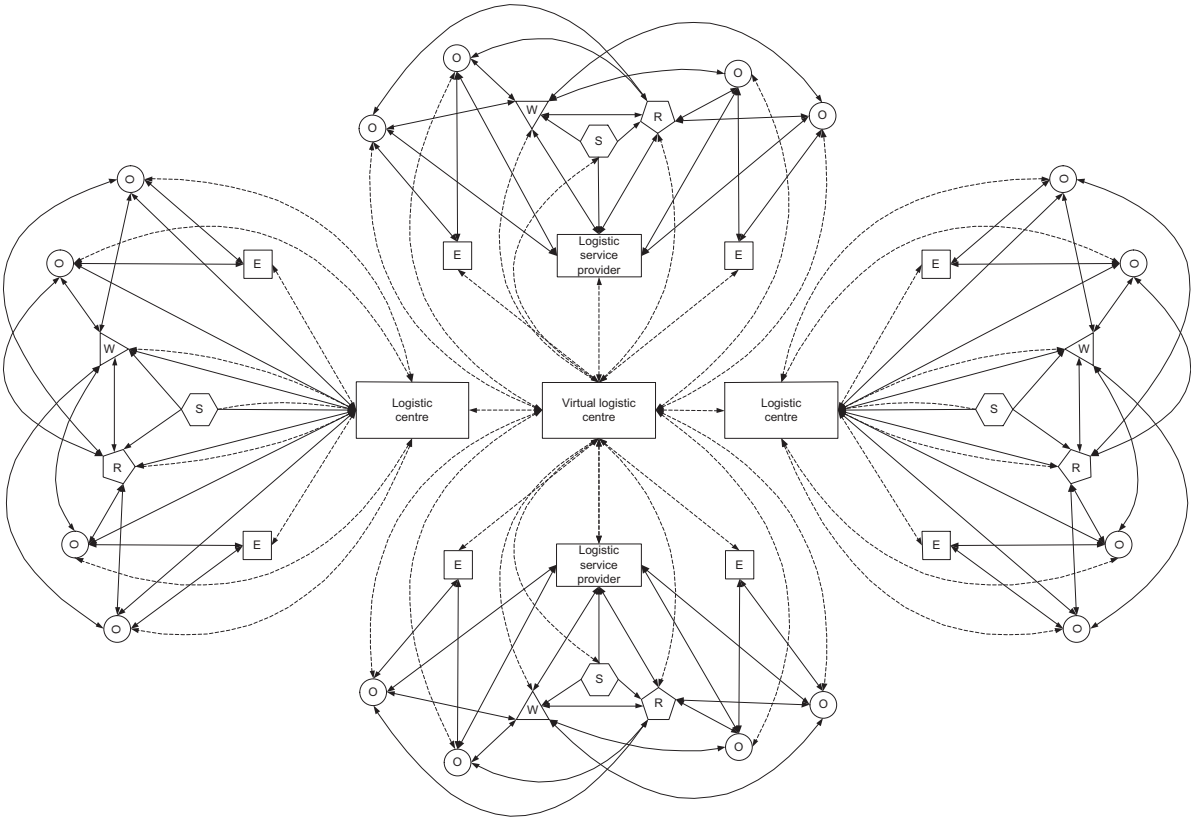


Fig. 1. General structure of a technical inspection and maintenance system Legend dashed lines: information flow, continuous line: material flow, E: expert, S: supplier, O: object, R: repair facility, W: warehouse.

The main parameter of the system is the path matrix $L = [l_{ij}]$, which shows the distances between the system elements. In this case, the path matrix is an integrated matrix, built up from several submatrices. The submatrices are defined by the number of elements in the system.

The assignment matrix $Y = [y_{ij}]$, is one of the main output parameters of the model, where $y_{ij} = \begin{cases} 1 \\ 0 \end{cases}$, the system elements are bound: 1 or not: 0.

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

3.1. Objects

The main parameters of the objects are:

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

In some systems, where human life is endangered, like at the elevators, the number of the technical inspections and maintenance events could be prescribed by the maintenance plan or even law or government regulations. In this model, the maintenance events and the inspections handled the same; there is no difference between them from the model point of view. The maintenance events cannot happen in an arbitrary period. There is a minimal time period ($\tau^{(m)}$) which has to be defined for every object after that the next maintenance task could be performed.

$$\tau^{(m)} = \left[\tau_i^{(m)}\right]_{i=1\ldots p}. \tag{1}$$

3.2. Experts

The main parameters are the following:

- s: is the number of experts, constant in most cases and also in this model, but it could change in some systems, for example, in the case of:
  - Expert leaving the job.
  - Employment of an expert.
  - The number of experts decreasing.
- P: is the performance of the experts; it shows how many actual maintenance tasks are performed by the expert.

The time required to travel between object i and j:

$$\tau_{i,j} = \frac{l_{i,j}}{v}; \quad \begin{matrix} i = 1 \ldots p \\ j = 1 \ldots p, \end{matrix} \tag{2}$$

where:
- $l_{i,j}$: is the distance between the object i and j.
- p: is the number of the objects.
- v: is the average speed of the expert.

3.3. Constraints

The interval of the inspections has to fulfill the constraint

$$\tau_i^{(m)} * (\varepsilon_i - 1) \leqslant \vartheta, \tag{3}$$

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

So all the examinations have to fit in the given 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, therefore the maintenance times could be shortened.

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

$$P_{i\,min} < P_i < P_{i\,max}, \tag{4}$$

The cycle time (τ_{max}) – usually it is one day, or a 8 h shift at countrywide or regional maintenance systems – is also a constraint, the expert visit the objects, do the inspection and/or maintenance and return to his base location in one cycle:

$$\tau^{(t)} = \tau_{0,1}^{(f)} + \tau_1^{(k)} + \sum_{i=2}^{c^t}\left(\tau_i^{(k)} + \tau_{i-1,i}\right) + \tau_{q,0}^{(f)} < \tau_{max}, \tag{5}$$

where:

– τ^t: is the interval during which the expert starts from his base location, visits the objects and returns:

$$\sum_{i=1}^{T}\tau_i^{(t)} \leqslant \vartheta, \tag{6}$$

where:

– T: is the number of cycles in the ϑ interval.
– τ_{max} time interval of a cycle.
– $c^{(t)}$: the number of objects to be visited in the cycle t.
– $\tau_{0,1}^{(f)}$: the travel time to the first object from the starting location.
– $\tau_{q,0}^{(f)}$: the travel time from the last object (q) to the expert base location.
– $\tau_i^{(k)}$: the average inspection time of the object i.

The set of objects defined which has to be inspected by the expert c can be:

$$O_c := \{o_i | y_{s,i} = 1;\ i = 1 \ldots p\}. \tag{7}$$

Thus the performance of an expert is:

$$|O_c| = P_c, \tag{8}$$

and the objects to be inspected in one cycle (t), the subsets of the O_c,:

$$O_c^{(t)} \subseteq O_s, \tag{9}$$

where:

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

$$o_p \in O_i; \quad o_q \in O_i; \quad o_p < o_q \text{ where } t_{o_p} < t_{o_q}, \tag{10}$$

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.

$$\left|O_c^{(t)}\right| = c_c^{(t)} \leqslant P_c, \tag{11}$$

$$\bigcup_{t=1}^{T} O_c^{(t)} = O_s, \tag{12}$$

and

$$\bigcup_{s=1}^{p} O_s^{(t)} = O. \tag{13}$$

The expert performs more than one inspection on an object, so the object is counted in the sets defined at (11) as many times as the number of inspection to be performed (Figs. 2 and 3).

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

$$d\left(o_i; o_j | o_i \in O_p^{(t)}; o_j \in O_q^{(t)}\right) = p - q, \tag{14}$$

so based on the constraint in Eq. (2):

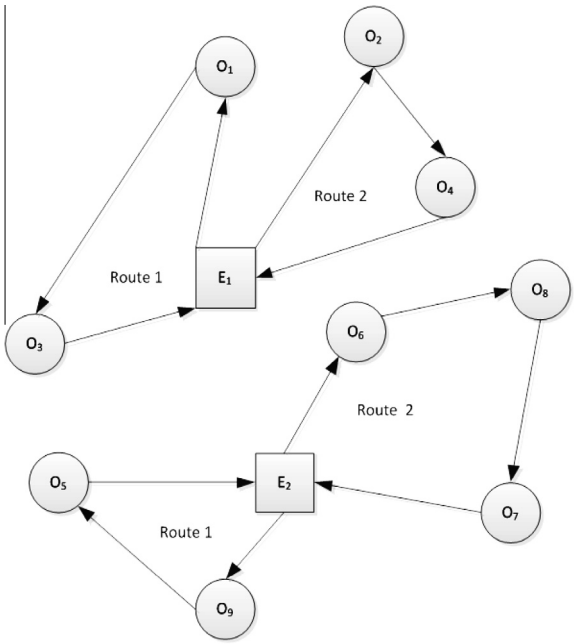


Fig. 2. A simple example of multiple routes with only one inspection at any object.

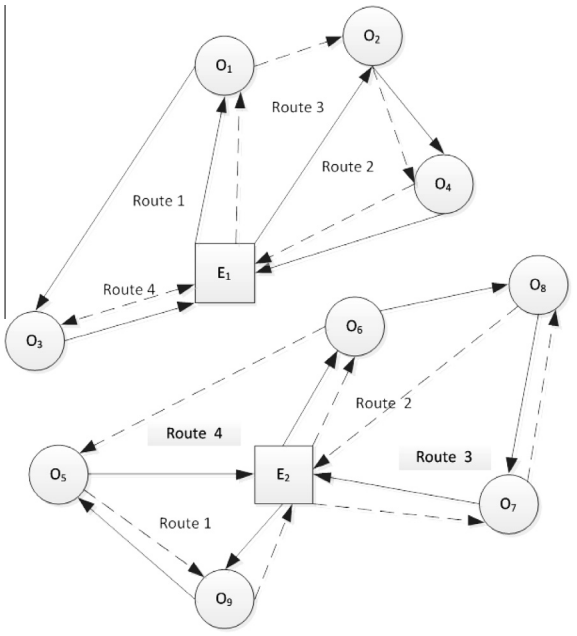


Fig. 3. A simple example of multiple routes with two inspections at the objects.

$$min\left\{d\left(o_i;o_j\middle|o_i\in O_p^{(t)};o_j\in O_q^{(t)}\right)\right\}\geqslant\tau_i^{(m)}. \tag{15}$$

So the path traveled by the expert i in a cycle t can be described as:

$$l_i^{(t)}=l_{0,O_i^{(t)}(1)}+\sum_{c=1}^{|O_i^{(t)}|-1}\left(l_{O_i^{(t)}(c),O_i^{(t)}(c+1)}\right)+l_{O_i^{(t)}\left(|O_i^{(t)}|\right),0}, \tag{16}$$

and the distance path traveled by the expert i can be described as:

$$l_i^{(T)} = \sum_{t=1}^{T}\left[l_{0,O_i^t(1)} + \sum_{c=1}^{|O_i^{(t)}|-1}\left(l_{O_i^{(t)}(c),O_i^{(t)}(c+1)}\right) + l_{O_i^{(t)}(|O_i^{(t)}|),0}\right] = \sum_{t=1}^{T} l_p^{(t)}. \tag{17}$$

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

$$C^{(S)} = \left[\sum_{j=1}^{s}\left(\sum_{t=1}^{T} l_j^{(t)}\right)\right] * c_u + \left[\sum_{j=1}^{s} P_j\right] * c_v, \tag{18}$$

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

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

$$C^S \rightarrow min, \tag{19}$$

the expenditures have to be minimal.
Constraints are described in the following equations: (3)–(6).

4. Literature on the multi-depot multiple traveling salesman problem

The assignment problems of the technical inspection and maintenance systems, discussed here, are close to the multiple traveling salesman problems (mTSP) [11,12]. Within the area of the mTSP, it is called the fixed destination multiple depot multiple traveling salesman problem (mdmTSP), but the solutions existing in the literature did not using sub-tour construction. Compared to the general TSP or mTSP problems this area is poorly researched. Only the newest studies deal with this field and they apply such heuristic techniques as:

- Agent-based modeling with probability collectives [13]: solve the mTSP with a multiagent model, where they used probability collectives, collective memory, and stochastic methods, this algorithm utilize simple inserting, swap and elimination heuristics. The algorithm was tested on two simple cases using three agents and fifteen nodes.
- Genetic algorithm [14]: they use a two phase algorithm to solve the problem. In the first phase, the problem was reduced problem into more single center problems from a multi-center problem. In the second phase, they using a genetic algorithm to solve the problems individually. The algorithm was tested up to 4 agents and 99 nodes.
- Ant colony algorithm [15]: here the ant colony algorithm developed by Marco Dorigo [16] was applied and general mTSP model was solved. In this method artificial ants search the solution in a problem tree imitating the foraging behavior of real ants. The result is compared with the solution of the Lingo 8 and the examination shows that the ant colony heuristic algorithm is far faster and it can give a better solution in many times. The algorithm was tested up to 5 agents and 40 nodes only.

The studies of optimization of complex logistics systems can be split into two main streams or areas. The first is using various metaheuristic optimization techniques [17] while the second focuses on simulation methods. However, simulation is a useful tool to optimize systems [18], but the execution time can be very high due to the complexity of these systems, especially in case of global sized, virtual systems [19].

The developed methods in the literature were tested on few nodes only, and they do not include the constraints of technical inspection and maintenance system which are mainly unique to this area. In real life logistics there are systems containing over 1000 nodes or rarely but there are systems containing over 10,000 nodes.

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

5.1. Evolutionary programming

The evolutionary programming is mainly used on heavily constrained problems. The evolutionary programming method is handling a population like the other evolutionary methods, but the problem representation is free, there are no limitations like at the genetic algorithm where bit vectors used to describe the individuals [20]. The problem modeled as it allows, or is it easy to use in computer solution.

In the computer solution, first the random generator, the input data and then the first population are initialized. There are two cases can be distinguished in heavily constrained problems:

- The randomly generated individual is not valid: it violates one or more constraints.

– The individual is in the feasible region: but this is the rare case.

There are various methods to get a valid individual from simply disposed invalid individuals to retain the integrity of the individual with special operators. But using penalty function is the simplest solution. In the penalty function, the preferred solution can be easily emphasized by the penalty values.

After the creation of the initial population, it has to be copied into a temporary population. And so the mutation operators run on this population. As usual, the high impact mutations have less chance to run, the low impact operators have a bigger chance. After the mutation finished, the mutated fitness value of the individuals has to be computed, and then the survivor individuals to the descendant population have to be chosen, which happens with a tournament. There is a simple way to perform the tournament is to choose two random individuals one individual from the original and one individual from the mutated population, and that will survive which has less (or bigger if the fitness not normalized) fitness value, it has to be repeated this until the new population is not filled.

At the evolutionary programming, in most cases, there is no crossover. In some solutions, there is no meaning of this, because it creates non valid individuals. So from the biological point of view this is not evolution of one species but the evolution of many but this naming convention is common for the integrity of evolutionary methods.

5.2. The problem representation in the proposed evolutionary programming algorithm

The algorithm we developed solves the fixed destination multiple depot multiple route multiple traveling salesman problem and optimizes the number of salesmen in one phase. It can be used for large or very large problems. The explanation of the definition is as follows; multiple salesmen: the experts; multiple depot: all the experts have different locations; multiple route: all the experts do more than one round routes; 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 has not widely used in genetic algorithm, but it could be simply implemented in the evolutionary programming. In these days, the usage of this technique is spreading solving more and more complex problems.

The multi chromosome technique uses more than one chromosome, unlike the genetic algorithm, to solve complex problems. Now there are even self-adapt methods based on multi chromosome technique like [21], where the cutting stock problem solved using multiple chromosomes. In [22], four chromosomes used to represent the input and output fuzzy sets of a proportional-plus-derivative fuzzy logic controller. In [23] multi chromosome method used to localize and quantify damage in truss structures.

The advantage of the multi chromosome technique is reducing the size of the search space. The problem approach of this modeling technique is similar to the characteristic of the problem so it can be more problem specific, therefore more effective [24].

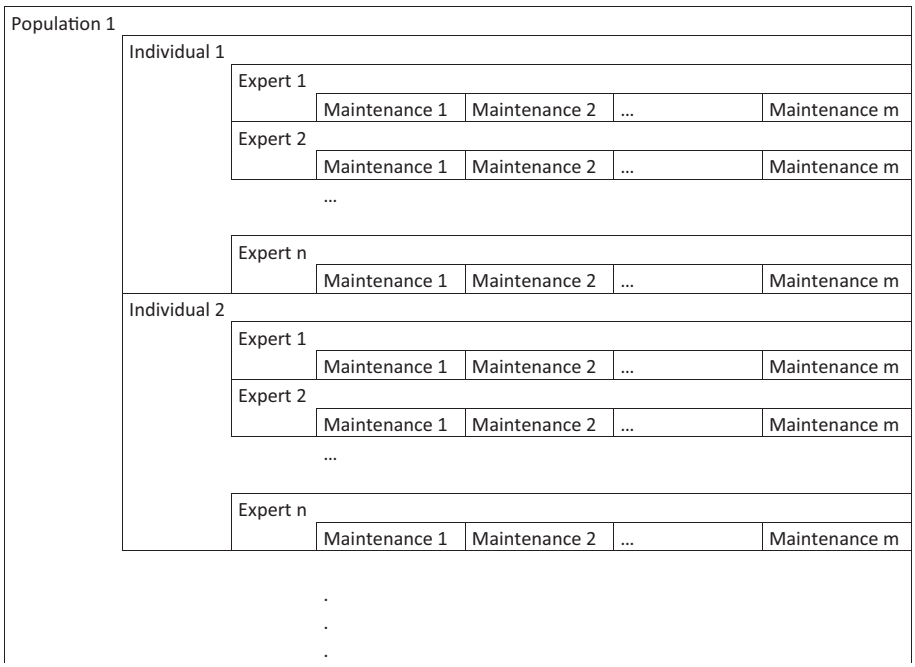


Fig. 4. The cascaded data structure of the optimization.

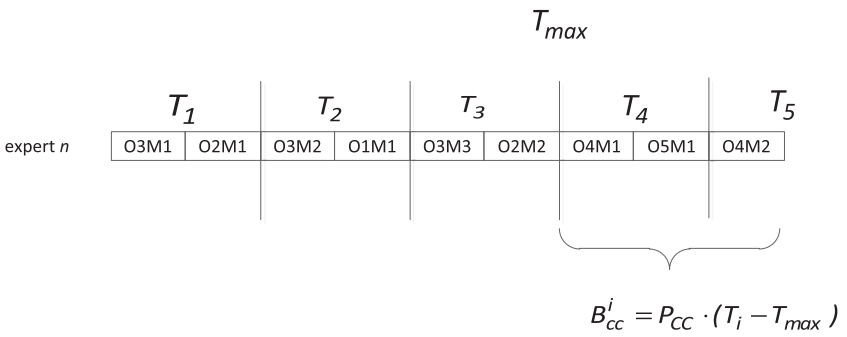


Fig. 5. Number of cycle penalty function (Legend: for example O4E1: maintenance/examination 1 of the object 4.).

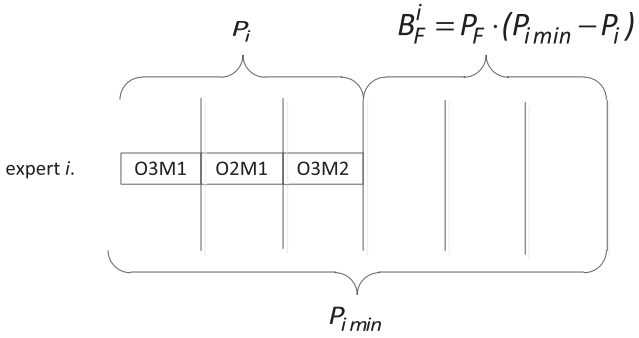


Fig. 6. Few penalty function.

The data structure of the optimization is built as described on Fig. 4. The biggest container is the population which consists of a defined – constant – number of individuals, which is an input parameter of the optimization.

First the algorithm initializes the constants, like maximum number of cycles, penalty constants, expert performance values, number of the individuals in a population and maximum tour length. It loads the data files of the experts that are defining the number of experts and their location. It defines the number of the objects and its location, and initializes maintenance data. Then it creates the first population filled with randomly generated individuals. It is a top down method:

- First generates container classes of individuals.
- Then generates container of experts and insert it into the container of the individuals.
- Then generates maintenance events in a randomly generated order and fills the chromosomes with them.

So every expert represented by a chromosome that is why the algorithm called as a multi chromosome algorithm. Then the algorithm calculates the fitness values of experts and applies the penalty functions. Finally it applies the global penalty functions and calculates the cumulated fitness value.

5.3. Penalty functions

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

- Local: the penalty function is applied to one of the experts.
- Global: the penalty function is given to the whole individual.

Some of the penalty functions have been described earlier in [25], but since then some of them reworked and the near penalty moved to the global penalties' group as we introduced the one object-one expert (Fig. 3) and the one object – multiple experts (Fig. 10) assignment model.

5.3.1. Local penalties

There are three different local penalty functions:

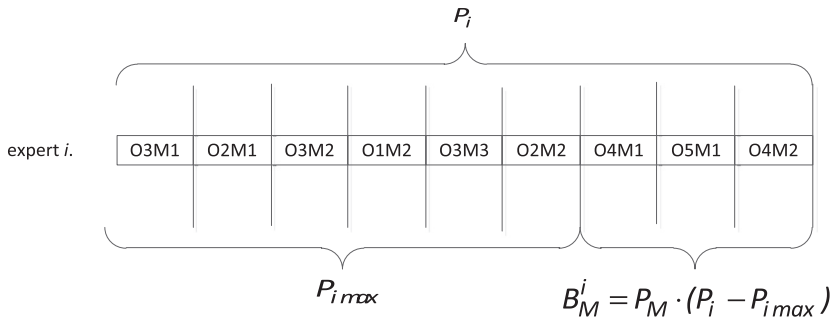


Fig. 7. More penalty function.

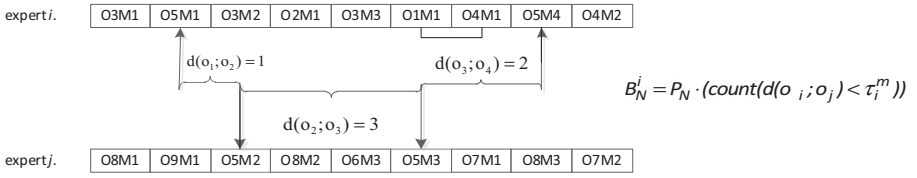


Fig. 8. Near penalty function.

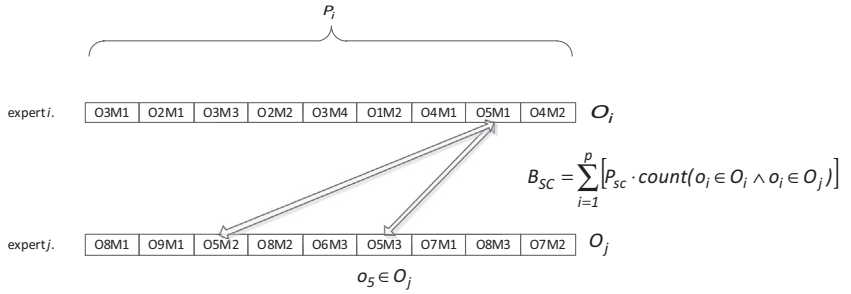


Fig. 9. Scatter penalty function.

– Number of cycles penalty: when the expert do more route cycles than allowed (Eq. (6)).
– Few penalty: the expert has to get a minimal number of maintenance events (Eq. (4)).
– More penalty: the expert cannot get more maintenance events than his maximum capacity (Eq. (4)).

The explanation of the different few and more penalty functions are the following:

In real life situations the employment of an expert with fewer examinations than the defined minimum is not economical, so the penalty value or the growing of the penalty value is bigger than the more penalty function. For example, the few penalty function is exponential, and the more penalty function is only linear, or sometimes a few penalty function gives is a death penalty value. The two constraints cannot be violated simultaneously so in the application it can be solved as one procedure.

Number of cycle penalty function of the expert i (Fig. 5):

$$B_{CC} = P_{CC} \cdot (T_i - T_{max})| \text{ if } T_i > T_{max}, \tag{20}$$

where:

– P_{CC}: is constant, the penalty value of the cycle count violation.
– T_i: is the cycles needed by the expert i.
– T_{max}: is the maximum allowed cycles (6).

It is applied when the experts runs more round route (one round route is one cycle) than the given maximum. For example the examined period (ϑ) is 365 days, then it has to be lesser or equal than 365 cycles (Eq. (6)). In some systems

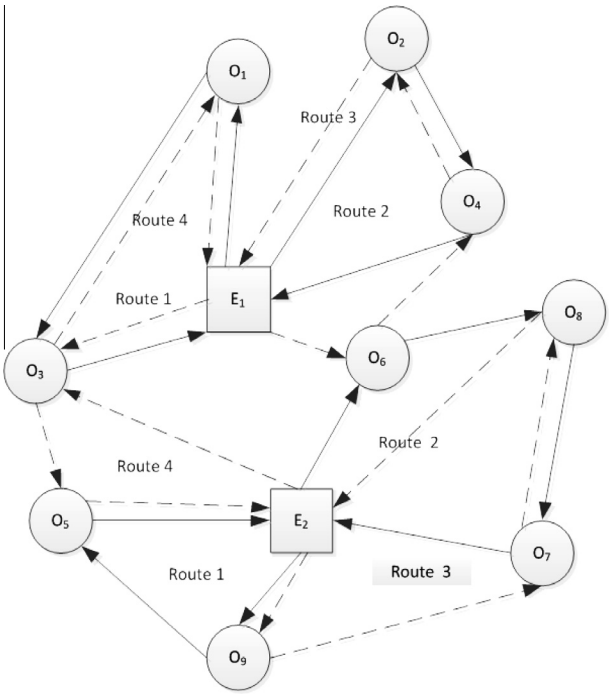


Fig. 10. An example of multiple routes with two inspections at any object without scatter penalty.

the violation of this constraint can be physically limited which justify the "death penalty" when the value of the penalty function is infinite.

Few penalty functions of the expert i (Fig. 6):

$$B_F = P_F \cdot (P_{i\,min} - P_i)| \text{ if } P_{i\,min} > P_i, \tag{21}$$

where:

- P_{FP}: is constant, the penalty value of the 'few maintenance' violation.
- P_i: is the actual performance of the expert i, the number of the maintenance events performed by the expert.
- $P_{i\,min}$: is the minimal allowed inspection count of the expert i (4).

The more penalty function of the expert i (Fig. 7):

$$B_M = P_M \cdot (P_i - P_{i\,max})| \text{ if } P_i > P_{i\,max}, \tag{22}$$

where:

- P_{MP}: constant, the penalty value of the more maintenance violation.
- P_i: the actual performance of the expert i, the number of the maintenance events.
- $P_{i\,max}$: is the maximum allowed inspection count of the expert i (4).

In some systems this could be regulated by law. In this case, the violation of the more penalty is not an economic issue and the usage of death penalty is justifiable.

5.3.2. Global penalties

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

- Near penalty: the maintenance events of one object cannot be arbitrarily close to each other (Eq. (1)).
- Scatter penalty: this applied when maintenance events of an object are scattered among several experts.
- Number of expert penalty: the employment of the expert has a fixed cost in this model. The algorithm tries to minimize the number of employed experts due to this penalty functions.

The near penalty function (Fig. 8) is the following:

$$B_N = P_N \cdot \left[count\left(d(o_i; o_j) < \tau_i^m\right)\right], \tag{23}$$

where:

- P_N: is constant, the penalty value of the near maintenance violation.
- $d(o_i; o_j)$: is the distance of examination i and j.
- τ_i^m: is the minimal distance of the examinations at the given object.
- $count()$: is counting when the condition is true.

The penalty function is taking account of the scattered examinations when the scatter penalty is not active.

In some systems, like the maintenance systems of the elevators, the minimal time distance of the examinations is controlled by government regulations.

Scatter penalty function is the following (Fig. 9):

$$B_{SC} = \sum_{k=1}^{p} [P_{SC} \cdot count(o_k \in O_x \wedge o_i \in O_y)], \tag{24}$$

where:

- P_{SC}: is constant, the penalty value of scattered maintenance events.

If the scatter penalty switched off, the algorithm is not forced to assign every maintenance event of one single object to one expert but will be distributed between experts (Fig. 10).

Number of expert penalty function, counts the experts who have inspection assigned:

$$B_S = count(P_i \neq 0|_{i=1..s}) \cdot P_{EC}, \tag{25}$$

where:

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

Then the algorithm enters into the main loop of the optimization and copies the population individuals into a temporary population in order of their fitness, so the individual with the best fitness value 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 result in faster algorithm.

The calculation of the fitness of the individual is:

$$F = C^{(S)} + B_s + B_{CC} + B_{SC} + B_F + B_M + B_N, \tag{26}$$

where:

- F: fitness of the individual.
- $C^{(S)}$: is the cost of the route traveled by the experts (Eq. (18)).
- B_S: is the cost of the employment of the experts (Eq. (25)).
- B_{CC}: is the output of the number of cycles penalty function (Eq. (20)).
- B_{SC}: is the output of the scatter penalty function (Eq. (24)).
- B_F: is the output of the few penalty function (Eq. (21)).
- B_M: is the output of the more penalty function (Eq. (22)).
- B_N: is the output of the near penalty function (Eq. (23)).

The target of the optimization application:

$$F \rightarrow min. \tag{27}$$

5.4. Mutation operators

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

- Inner expert mutation operators, working on one chromosome.
- Cross expert mutation operators, working on two different chromosomes.

In the next chapters introductions of six mutation operators are treated, but in fact those are combinations of two gene swap operators (local, global) and an insertion (local) and a contraction operator (global), thus their number is four with

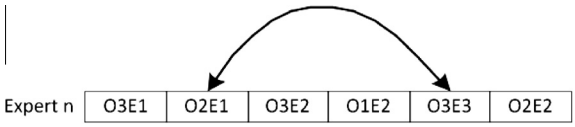


Fig. 11. Gene swap operator.

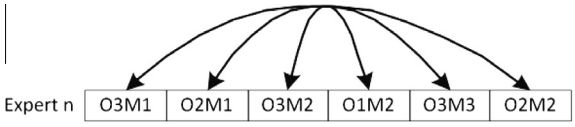


Fig. 12. Gene reversion operator.

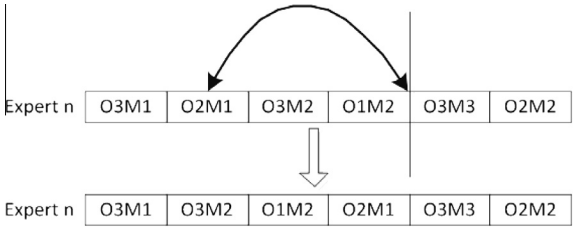


Fig. 13. Gene insertion operator.

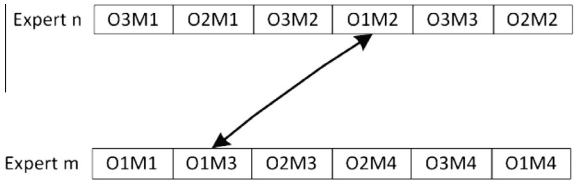


Fig. 14. Cross expert gene swap operator.

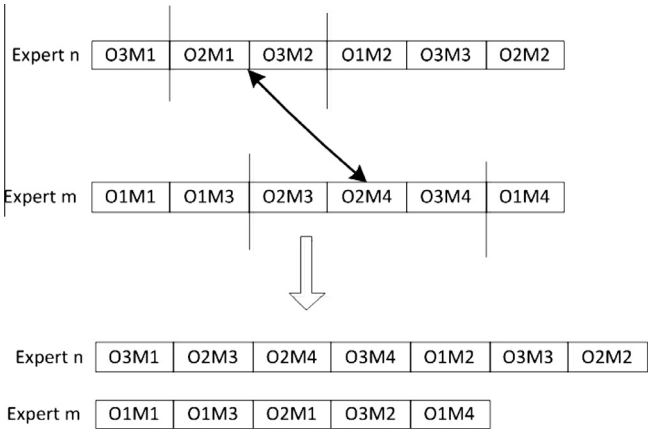


Fig. 15. Cross expert gene sequence swap operator.

combinations. The mutation operators have been described in [25], although we have introduced a new chromosome contraction operator according to Fig. 17.

5.4.1. Inner expert mutation

There are three types of inner expert mutation operators:

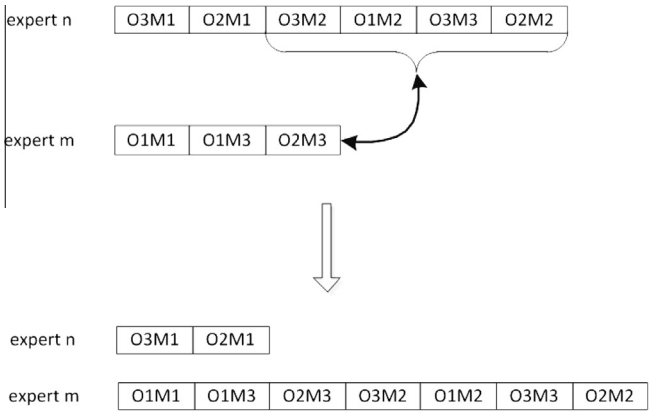


Fig. 16. Cross expert chromosome contraction Type 1.

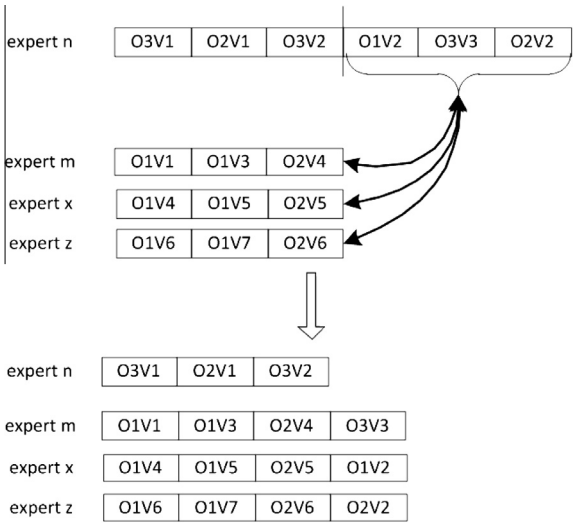


Fig. 17. Cross expert chromosome contraction Type 2.

- Gene swap: two random genes are swapped (Fig. 11).
- Gene sequence reversion: the gene sequence is reversed between two random indexes (Fig. 12).
- Gene insertion operator: a randomly chosen gene is inserted into a randomly chosen position (Fig. 13).

5.4.2. Cross expert mutation

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

- Cross expert gene swap: swaps two genes between experts. The second expert and the position of the other gene are chosen randomly (Fig. 14).
- Cross expert gene sequence change: the algorithm is swapping a randomly chosen but continuous gene sequence with a randomly chosen experts' also randomly chosen gene sequence (Fig. 15).
- Cross expert gene contraction: where a 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 operator contracts a randomly chosen chromosome (expert n) by displacing random amount of 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. 16).
  - The second type of contraction operator is displacing random amount of genes from the end of a randomly chosen chromosome like the first operator, but it spreads the genes between randomly chosen chromosomes one by one (Fig. 17).

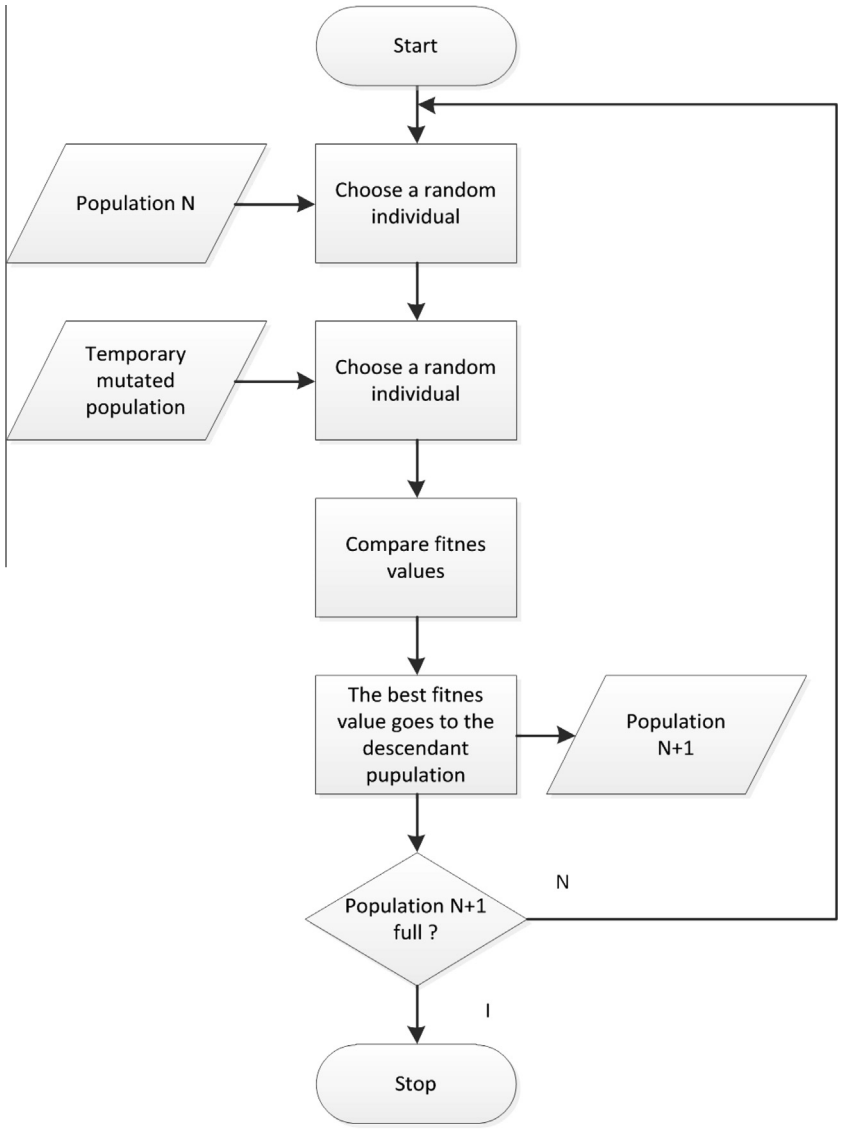


Fig. 18. Algorithm of the tournament process.

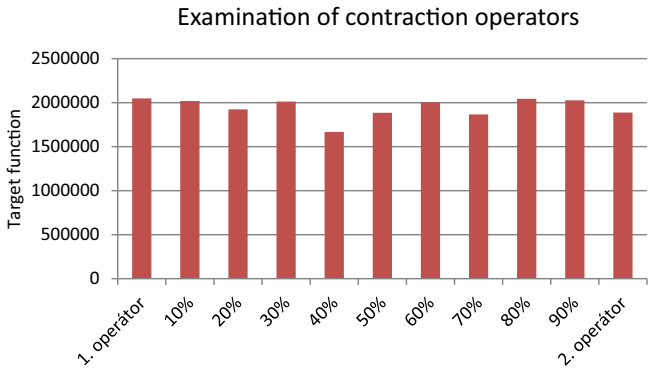


Fig. 19. First examination of contraction operators.

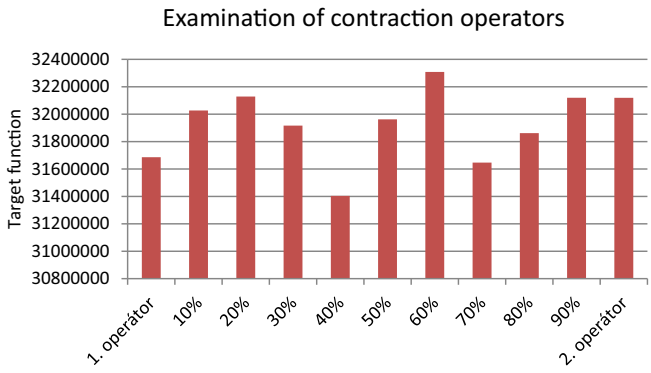


Fig. 20. Second examination of contraction operators.

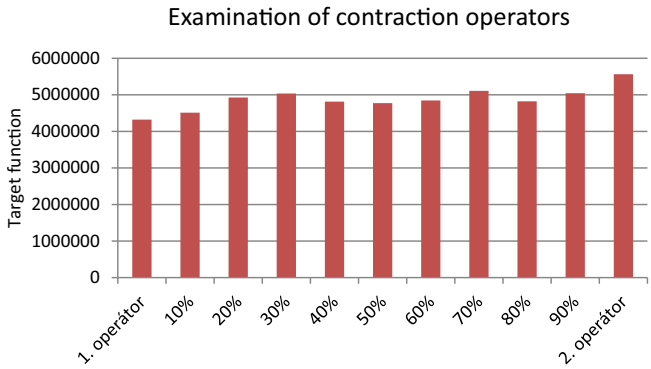


Fig. 21. Second examination of contraction operators.

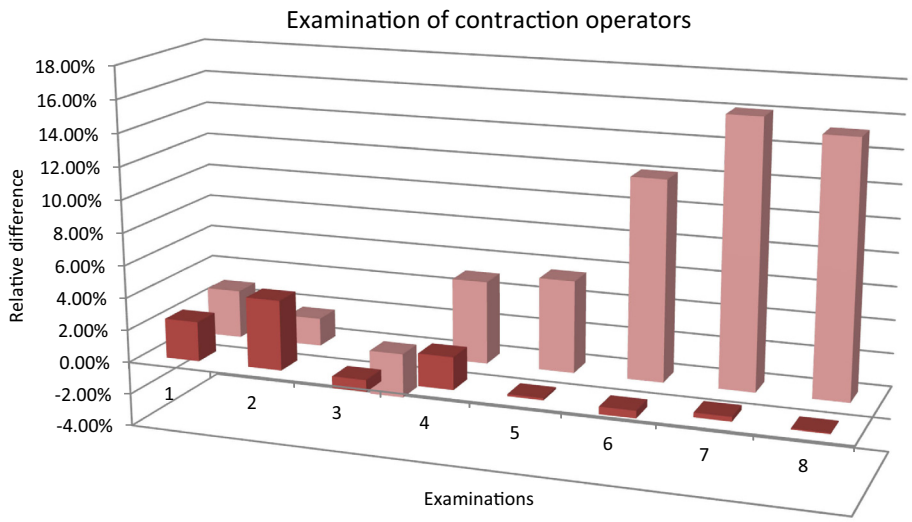


Fig. 22. Examination of contraction operators, with 40% second operator probability Legend first sequence: 1st operator + 2nd with 40% probability, second sequence: 2nd operator with 100% probability.

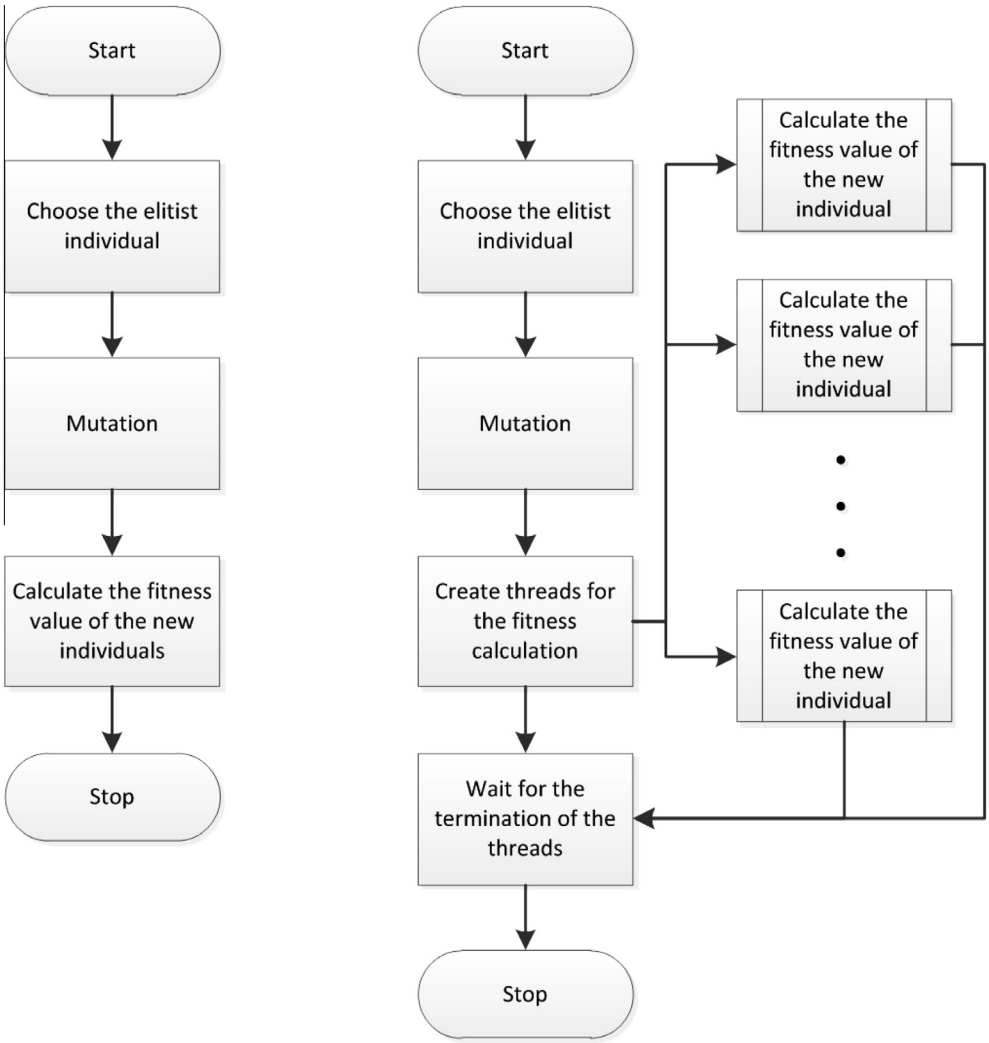


Fig. 23. Parallelization of the fitness calculation (a, serial execution, b, parallel execution).

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

5.5. Survivor selection

First, the algorithm searches for the best individual and copies 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 the algorithm randomly chooses two individuals (1 + 1 strategy), one from the original and one from the mutated population and the fittest wins. The winner individual will be copied into the next generation of the individual. This process is repeated until the next population is filled.

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

The tournament process (Fig. 18):

- Randomly chooses an individual from the original and another from the mutated population.
- The individual with the best fitness value is inserted into the descendant population.
- Repeats 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 a local optimum.

Table 1
Examination of parallelization with a small number of individuals.

No. of iterations	Instance (TSPLIB)	Number of nodes	Parallelized 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	11,849	No	12:01	
100	rl11849	11,849	Yes	10:03	19.57
100	usa13509	13,509	No	13:56	
100	usa13509	13,509	Yes	11:49	17.91

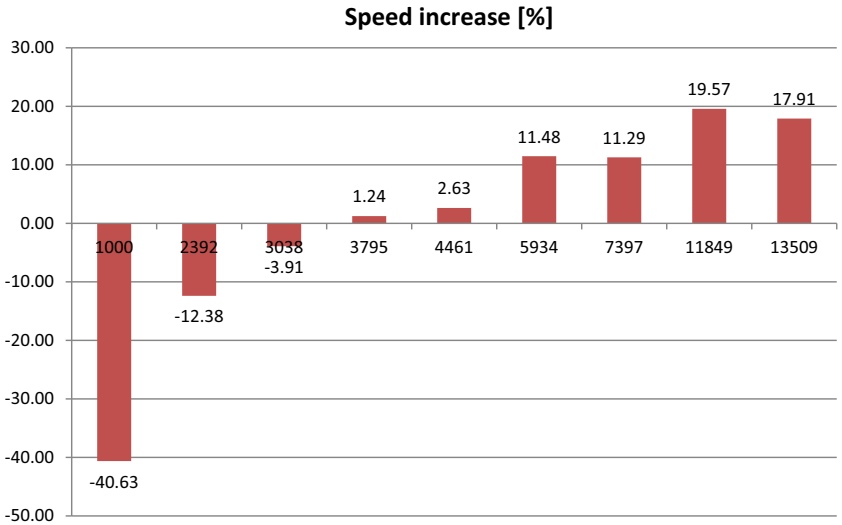


Fig. 24. Speed increase in the function of the number of nodes.

5.6. Examination of the two contraction operators

All the examinations in this chapter are based on the "ring" instance presented in Section 7.

The examination of the two contraction operators gives contradictory results. So the operators had to be analyzed through several runs with the same parameters.

First examination: 3 experts 48 nodes, number of examination 2–4 (Fig. 19).

The first examination: 3 experts, 48 nodes, the number of examinations is 5–10.

The second examination: 3 experts, 48 nodes, the number of examinations is 10–15 (Fig. 20).

The examination shows that in some cases, when the second contraction operator is also used with 40 percent probability, it gives better results. So another examination was performed with the 40 percent probability of the second operator with increasing problem size (Fig. 21).

The examination shows that the usage of the second operator with 40% probability is giving better result just in some random cases (Fig. 22) and the improvement is 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 that the running times grow exponentially so the parallelization of the algorithm or parts of the algorithm is obvious to run on multiprocessor computers or even computing clouds.

Table 2
Examination of parallelization with a large number of individuals.

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	11,849	No	24:02	
50	rl11849	11,849	Yes	19:15	24.85
50	usa13509	13,509	No	28:18	
50	usa13509	13,509	Yes	22:30	25.78

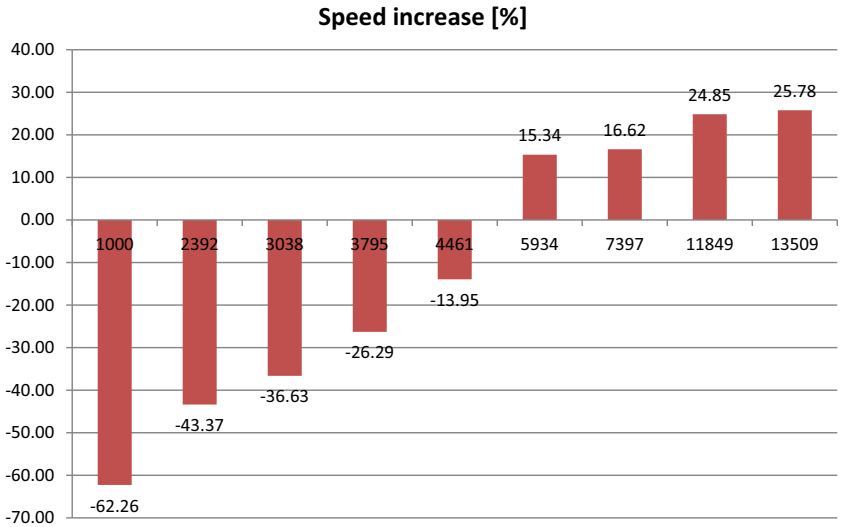


Fig. 25. Speed increase in the function of node at larger problems.

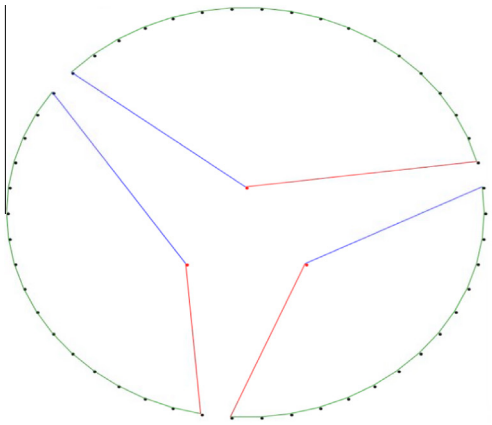
The algorithm analysis shows that the mutation and the fitness calculation are the two most used and most time consuming procedures. The mutation process due to the multi-chromosome design is less suitable for parallelization than the single chromosome because the global mutation operators process the entire individual, so the locks of the individuals' memory area could slow the process, but it can profit from the parallelization too. However, the fitness calculation can easily parallelize because it does not do any data modification.

The gain of the parallelization is not obvious. The operating system assigns the threads to the available processor cores and schedules 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.

6.1. Examination of the parallelization

The examination of the parallelization (Fig. 23) needs large scale problems, so the TSPLIB library was chosen for testing instances [26]. The examination methodology was:



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

Fig. 26. Test instance with three experts.

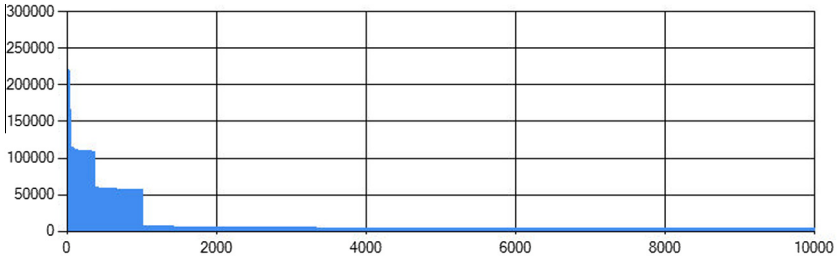
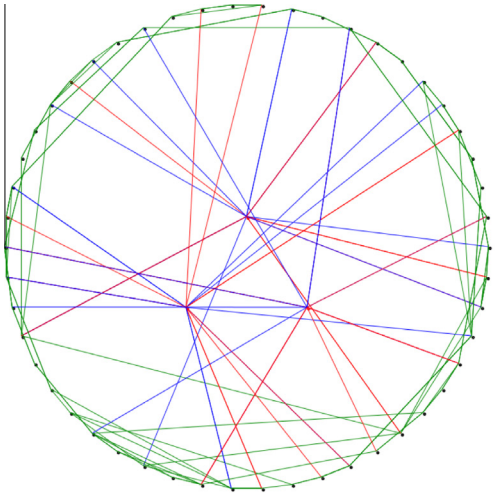


Fig. 27. Convergence of the solution.



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

Fig. 28. Test instance with three experts.

– Random generator initialized with the same value.
– Same TSPLIB data.
– Equal number of iterations.

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

The diagram (Fig. 24) 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

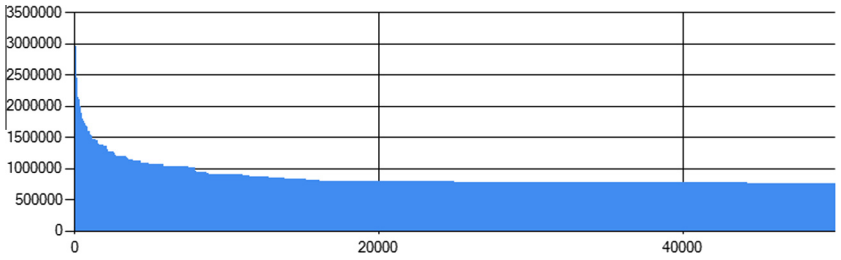


Fig. 29. Convergence of the solution.

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

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

		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

with these input parameters and settings. The results also show that the gain with four processor cores is not exceeding 20%, but the performance gain of the parallelization is obvious on large instances.

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

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 (Fig. 25). The speed gain threshold limit is doubled, compared to the first test. It is about 5200 nodes in this case. So it is obvious that solving large or very large scale problems often requires computing clouds or specialized GPU farms.

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 needs not to test and refine the algorithm for further improvements and also for solving real life problems.

7.1. Test instance with three experts

This test instance uses three experts, and 48 nodes, all nodes must be inspected only once so the results can easily be proved by the naked eye (Figs. 26 and 27).

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) (Figs. 28 and 29).

Table 5
Calculating the number of iterations.

Iteration number	Population size	Examined number of individuals
10,000	10	100,000
5000	20	100,000
4000	25	100,000
2000	50	100,000
1000	100	100,000
500	200	100,000
333	300	99,900
250	400	100,000
200	500	100,000
166	600	99,600
142	700	99,400
125	800	100,000
111	900	99,900
100	1000	100,000

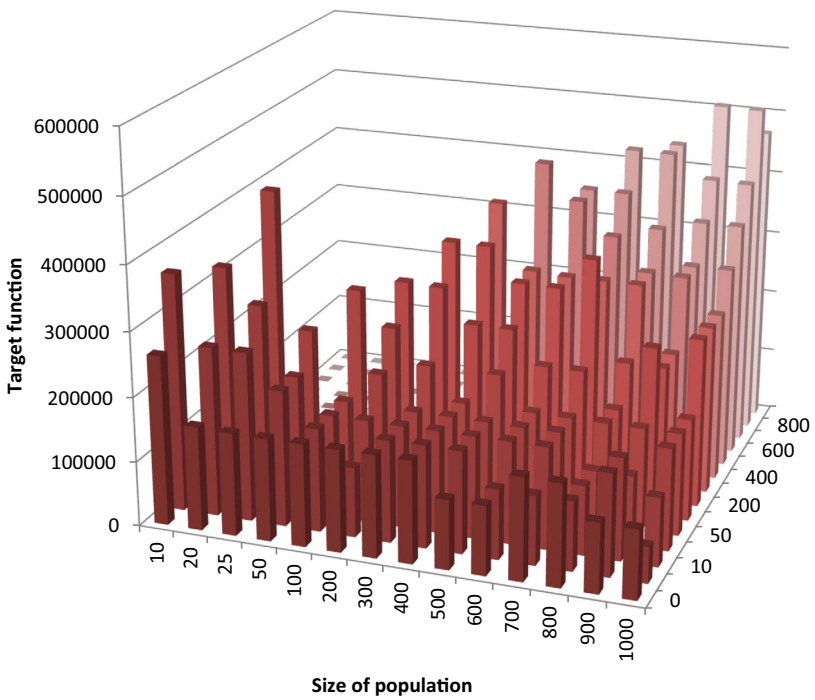


Fig. 30. Changes of the target function in the function of the population size and the number of random individuals.

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

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

		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						

8. Sensitivity analysis

8.1. 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 maintenance events. The results of the sensitivity analysis, the best target function value marked with green color (see Tables 3 and 4).

If the problem is examined in equal size of the search space (Table 5):

The results show (Fig. 30) that the increase of the number of the randomly generated individuals degrades the obtained solution because the random individuals reduce the discovered search space at this small example. At greater problems, the effect of the random individuals is more reasonable, at the examinations showed the number of random individuals is about 2% of the problem size.

These results (Tables 6 and 7 and Fig. 31) show the effect of the randomly generated individuals. The optimal ratio is in the area of 0–3%. The test shows some unconventional results also, but these are random perturbations. The test run shows

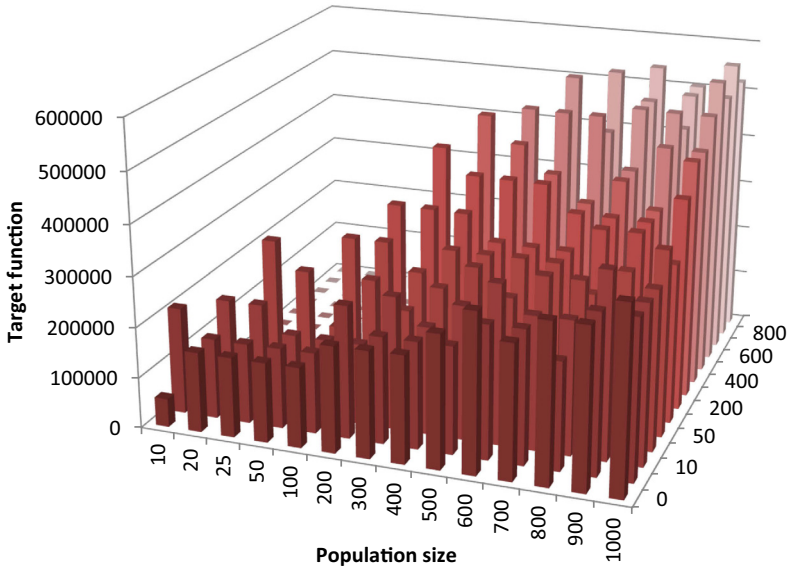


Fig. 31. Changes of the target function in the function of the population size and the number of random individuals.

that it is not worth to give much higher number of individuals than the number of nodes because, in that case, the runtime could grow considerably. If the number of iterations is decreased, the effect of the selection is not prevailing, 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 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 evolutionary method, which can integrate several constraint functions relatively easily. The analysis of the test examples shows that, for large scale problems, the usage of a high computation capability computer cloud is highly recommended.

10. Further improvements

The algorithm shows great potentials toward further improvements. Firstly, improvements in the method we used himself:

- New mutation methods.
- New survivor selection methods.

Secondly, improvements to increase speed of the algorithm:

- Massive parallelization in a computing cloud.
- Examine the implementation possibilities on graphics processing units (GPU) or a GPU cluster.

Thirdly, try to solve the problem with other optimization methods, like the firefly algorithm which is a member of the family of the particle swarm optimizations, and compare it with the developed evolutionary algorithm.

Acknowledgements

The research was supported by the TÁMOP 4.2.4.A/2-11-1-2012-0001 priority Project entitled "National Excellence Program – Development and operation of domestic personnel support system for students and researchers, implemented within the framework of a convergence program, supported by the European Union, co-financed by the European Social Fund. The research was supported also by the Hungarian Scientific Research Fund OTKA T 109860 project and was partially carried out in the framework of the Center of Excellence of Innovative Engineering Design and Technologies at the University of Miskolc.

References

[1] J. Levitt, The Handbook of Maintenance Management, Industrial Press Inc, 2009. ISBN 978-0-8311-3389-4, p. 477.
[2] A. Király, J. Abonyi, Optimization of multiple traveling salesmen problem by a novel representation based genetic algorithm, Intell. Comput. Optim. Eng. Stud. Comput. Intell. 366 (2011) 241–269, http://dx.doi.org/10.1007/978-3-642-21705-0_9PB.
[3] D. Achermann, Modeling, Simulation and Optimization of Maintenance Strategies under Consideration of Logistic Processes (PhD. thesis), Swiss Federal Institute of Technology, Zurich, Diss. ETH No. 18152, 2008.
[4] L. Cser, J. Cselényi, M. Geiger, M. Mäntylä, A.S. Korhonen, Logistics from IMS towards virtual factory, J. Mater. Process. Technol. 103 (1) (2000) 6–13, http://dx.doi.org/10.1016/S0924-0136(00)00412-X. ISSN 0924-0136.
[5] M. Lam, J. Mittenthal, Capacitated hierarchical clustering heuristic for multi depot location routing problems, Int. J. Logistics Res. Appl. Leading J. Supply Chain Manage. (2013), http://dx.doi.org/10.1080/13675567.2013.820272.
[6] D. Chao, C. Ye, H. Miao, Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale TSPs, Tsinghua Sci. Technol. 12 (4) (2007) 459–465. ISSN 1007-0214 15/20.
[7] P.C. Pop, I. Kara, A.H. Marc, New mathematical models of the generalized vehicle routing problem and extensions, Appl. Math. Model. 36 (1) (2012) 97–107, http://dx.doi.org/10.1016/j.apm.2011.05.037.
[8] R. Nallusamy, K. Duraiswamy, R. Dhanalaksmi, P. Parthiban, Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics, Int. J. Nonlinear Sci. 8 (4) (2009) 480–487. ISSN 1749-3889 (print), 1749-3897.
[9] D. Chao, C. Ye, H. Miao, Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale TSPs, Tsinghua Sci. Technol. 12 (4) (2007) 459–465. ISSN 1007-0214 15/20.
[10] G. Mosheiov, Vehicle routing with pickup and delivery: tour partitioning heuristics, Comput. Ind. Eng. 34 (3) (1998) 669–684, http://dx.doi.org/10.1016/S0360-8352(97)00275-1.
[11] D.L. Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2006. ISBN 978-0-691-12993-8.
[12] A.E. Carter, C.T. Ragsdale, A new approach to solving the multiple traveling salesperson problem using genetic algorithms, Eur. J. Oper. Res. 175 (1) (2006) 246–257.
[13] A.J. Kulkarni, K. Tai, Probability collectives: a multi-agent approach for solving combinatorial optimization problems, Appl. Soft Comput. 10 (2010) 759–771, http://dx.doi.org/10.1016/j.asoc.2009.09.006.
[14] T.S. Bae, H.S. Hwang, G.S. Cho, M.J. Goan, Integrated GA-VRP solver for multi-depot system, Comput. Ind. Eng. 53 (2) (2007) 233–240, http://dx.doi.org/10.1016/j.cie.2007.06.014.
[15] S. Ghafurian, N. Javadian, An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems, Appl. Soft Comput. 11 (2011) 1256–1262.
[16] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004. ISBN 0-262-04219-3.
[17] D. Simchi-Levi, X. Chen, J. Bramel, Logic of Logistics: Theory, Algorithms, and Applications for Logistics and Supply Chain Management, Springer-Verlag, 2004. ISBN: 0387221999, 355 p.
[18] T. Bányai, Optimisation of U-shaped flexible manufacturing cells, in: B. Katalinic (Ed.), Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium "Intelligent Manufacturing & Automation: Focus on Theory, Practice and Education", Published by DAAAM International Vienna, Vienna, Austria, 2009, pp. 761–762. ISSN 1726–9679, ISBN 978-3-901509-70-4.
[19] Á. Bányai, Das virtuelle Logistikzentrum als Koordinator der logistischen Aufgaben, in: T. Bányai, J. Cselényi (Eds.), Modelling and Optimization of Logistic Systems – Theory and Practice, Published by the University Miskolc, 1999, pp. 42–50. ISBN 963 661 402 4.
[20] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third rev. and extended ed., Springer, 1996. 387 p.
[21] R. Hinterding, Self-adaptation using multi-chromosomes, in: Proceedings of the IEEE International Conference on Evolutionary Computation, Indianapolis, United States, 1997, pp. 87–91.
[22] N. Baine, A simple multi-chromosome genetic algorithm optimization of a proportional-plus-derivative fuzzy logic controller, in: Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society, IEEE Press, New York City, United States, 2008, pp. 1–5.
[23] J.D. Villalba, J.E. Laier, Localising and quantifying damage by means of a multi-chromosome genetic algorithm, Adv. Eng. Software 50 (2012) 150–157, http://dx.doi.org/10.1016/j.advengsoft.2012.02.002. ISSN 0965-9978.
[24] A. Király, J. Abonyi, Optimization of multiple traveling salesmen problem by a novel representation based genetic algorithm, Stud. Comput. Intell. 313 (2010) 141–151, http://dx.doi.org/10.1007/978-3-642-15220-7_12.
[25] L. Kota, Optimisation of Large Scale Maintenance Networks with Evolutionary Programming, DAAAM International Scientific Book, 2011, pp. 495–512. ISSN 1726–9687, ISBN 978-3-901509-84-1, Chapter 40. http://dx.doi.org/10.2507/daaam.scibook.2011.40.
[26] 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/>, accessed on 09.09.2013.