Parameter Tuning in an Evolutionary Algorithm for Commodity Transportation Optimization

Erik Dovgan, Tea Tušar and Bogdan Filipič, *Member, IEEE*

***Abstract*— Tuning parameters of an evolutionary algorithm is the essential phase of a problem solving process since the parameter values significantly influence the algorithm efficiency. A traditional parameter tuning approach finds a setting of parameter values that is then used for solving various problem instances. Clearly, such parameter values may not perform well on specific problem instances. This paper suggests finding several parameter settings which are suitable for specific problem instances. However, this is not aimed at the level of each individual instance, but rather for specific types of problem instances. A new problem instance can then be solved using the tuned parameter values for its type. We demonstrate the approach by tuning parameters of an evolutionary algorithm for commodity transportation optimization with very heterogeneous problem instances. Numerical experiments show that the procedure improves the algorithm performance. Moreover, the analysis of empirical results reveals that there exist relations between the tuned parameter values and that they vary over types of problem instances.**

I. Introduction

Evolutionary algorithms (EAs) are stochastic search and optimization procedures designed to solve a wide range of problems. They improve candidate solutions over generations using selection and variation operators. Their structure is rather general and their performance depends on the algorithm parameters. Implementing an EA to solve a specific problem therefore involves finding appropriate values of the algorithm parameters [3]. Determining parameter values is a hard task since there is not much knowledge about the effects of EA parameters on the algorithm performance [6]. Nevertheless, parameter values determine whether or not the algorithm will find an optimal or near-optimal solution [4]. Therefore, finding good parameter values is important even if the process requires a lot of additional resources.

Parameter values can be chosen before or during the algorithm execution. If the values are chosen before, the process is called parameter tuning. Usually, this is a very time-consuming process. Besides, parameter values are tuned on a limited number of tested problem instances. There is no guarantee that the algorithm with tuned parameter values will perform well on previously unseen problem instances [4]. Therefore, each instance requires its own setting of parameter values which is impracticable and in contrast to the traditional approach of using a single setting of parameter values for a problem.

This paper presents a way to improve the EA performance by determining specific parameter values for groups of problem instances, which represents a trade-off between the usage of problem specific parameter values and problem instance specific parameter values. This is done by first identifying several types of problem instances and then finding suitable parameter values for each type. A new problem instance can then be solved using the parameter values of the instance type it belongs to. This approach is illustrated empirically by tuning parameter values of an EA for a commodity transportation optimization.

The paper is structured as follows. Section II presents a short overview of the related work. The studied problem is described in Section III. An EA designed to deal with this problem is presented in Section IV. The idea of tuning parameters for groups of problem instances is presented in Section V. Numerical experiments and results are reported in Section VI and discussed in Section VII. Section VIII concludes the paper by summarizing the work and providing directions for future work.

II. Related Work

The issue of parameter setting has been relevant since the beginning of the EA research and practice. In this section we summarize recent work in this domain.

Eiben et al. [2] [4] emphasize that the parameter values greatly determinate the success of an EA in finding an optimal or near-optimal solution to a given problem. Choosing appropriate parameter values is a demanding task. There are two major forms of setting parameter values: parameter tuning and parameter control. In parameter tuning, parameter values are defined in advance and do not change during the algorithm execution. Parameter tuning is a time-consuming activity since the number of possible settings is very large. Besides, there are also other drawbacks, e.g., parameters are not independent but trying all combinations is impossible and the found parameter values are appropriate only for the tested problem instances. Moreover, Nannen et al. [6] state that another problem of parameter tuning is weak understanding of the effects of EA parameters on the algorithm performance.

On the other hand, parameter control changes parameter values during the algorithm run and can be deterministic, adaptive or self-adaptive. A deterministic parameter control changes the parameter values regarding predefined rules

Erik Dovgan is with the Department of Intelligent Systems, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia (phone: +386 1 477 3393; email: erik.dovgan@ijs.si).

Tea Tušar is with the Department of Intelligent Systems, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia (phone: +386 1 477 3462; email: tea.tusar@ijs.si).

Bogdan Filipič is with the Department of Intelligent Systems, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia (phone: +386 1 477 3352; email: bogdan.filipic@ijs.si).

without any feedback. An adaptive parameter control changes parameter values regarding predefined rules by taking into consideration the feedback. Self-adaptive parameter control evolves the parameter values since they are part of the chromosomes and thus exposed to variation [4].

Smit and Eiben [8] presented an overview and comparison of EA parameter tuning methods. In addition, they analyzed how beneficial tuning is even to experienced practitioners. Their experiments showed that there is no best value for each parameter, but there are wide ranges of good parameter values. The authors also showed that it is hard to consistently reach the areas with the best EA setup. Nevertheless, they concluded that using algorithms for tuning EA parameters does pay off in terms of EA performance, since the best guess of all tuning algorithms in their study greatly outperformed the best guess of a human user.

Diaz-Gomez and Hougen [1] started from the idea that the EA parameter values are not independent. Therefore, their values cannot be chosen arbitrarily but must meet the limitations, which are, for example, expressed as equations. They investigated the connection between the crossover probability, the mutation probability and the selection pressure. The connection was expressed as an equation defining a range of possible parameter values. The approach was tested on two problems and the parameter values of the best chromosomes met the equation requirement thus demonstrating that the parameter values for achieving better solutions are not independent.

III. Problem Description

In our case, parameter tuning was studied on an EA for a specific transportation optimization problem. The problem comprises transportation of heterogeneous cargo from location A to location B that has to be done in minimal time. The cargo consists of n_c kinds of commodities c_j with given amounts w_j, where $j = 1, \ldots, n_c$. The cargo is transported with a group of heterogeneous vehicles that always travel in a convoy.

Commodities are transported using n_v vehicles. Each vehicle v_i can transport the amount w_{ij} of commodity c_j, where $i = 1, \ldots, n_v$ and $j = 1, \ldots, n_c$. We call w_{ij} the capacity of vehicle v_i for commodity c_j. Vehicle v_i may not be able to transport commodity c_j. In this case, $w_{ij} = 0$. Although a vehicle can transport several kinds of commodities, it can transport only one kind of commodity at a time. All available vehicles must be used, unless location A runs out of all commodities that can be transported using a particular vehicle.

The route between locations A and B is predefined and does not change. It consists of a sequence of roads with defined lengths and speed limits. Additionally, speed limits for different types of roads are defined for each vehicle traveling loaded or unloaded. These data are used for calculating the traveling time of vehicle v_i between locations A and B when the vehicle is loaded, t_i^f, and between locations B and A when it is unloaded, t_i^e.

At location A, n_l loading points are available. The speed of loading commodity c_j at loading point l_k is a_{kj} commodity per hour, where $k = 1, \ldots, n_l$ and $j = 1, \ldots, n_c$. If the loading point l_k is not able to load commodity c_j, then $a_{kj} = 0$. At location B, n_u unloading points u_k are available for which the unloading speeds $b_{kj} \geq 0$, where $k = 1, \ldots, n_u$ and $j = 1, \ldots, n_c$, are given analogously. A loading point can load only one vehicle at a time. Consequently, at each point there is a queue of waiting vehicles. Vehicle loading continues with no interruption until the vehicle is full or location A runs out of the commodity loaded by the vehicle. The same holds for unloading points (unloading continues with no interruption until the vehicle is empty).

Some vehicles are equipped with a self-(un)loading device. The self-loading speed of vehicle v_i for commodity c_j is defined as a_{ij}^s and its self-unloading speed is defined as b_{ij}^s, where $i = 1, \ldots, n_v$ and $j = 1, \ldots, n_c$. If the vehicle self-(un)loading device is not able to (un)load commodity c_j, then $a_{ij}^s = 0$ ($b_{ij}^s = 0$). If the vehicle does not have a self-(un)loading device, $a_{ij}^s = 0$ ($b_{ij}^s = 0$) for all commodities c_j, $j = 1, \ldots, n_c$. Although a vehicle has such an equipment, its usage is not required, unless there is no loading or unloading point for the assigned commodity. If a vehicle uses its self-loading device for loading, it does not have to wait for the loading point to become available and it does not occupy the loading point when it is loaded. On the other hand, self-loading can be slower than loading at loading points, therefore it is not trivial to decide whether or not to use a vehicle's self-loading device. Analogous reasoning applies for self-unloading.

IV. An Evolutionary Algorithm for Commodity Transportation Optimization

An EA was designed to optimize the described commodity transportation. In the following subsections we describe the solution representation, their evaluation, and the problem solving procedure carried out by the EA.

A. Solution Representation

A solution of the transportation problem can be represented as a transportation plan P. Usually the commodity amounts exceed the capacities of vehicles, therefore transportation has to be done in several cycles, namely $P = P^1, P^2, \ldots$ In each cycle $P^k = (u_1^k, x_1^k, y_1^k, w_1^k), \ldots, (u_{n_v}^k, x_{n_v}^k, y_{n_v}^k, w_{n_v}^k)$, each vehicle v_i^k is assigned:

- a commodity u_i^k (it is 0 if location A runs out of all commodities that can be transported with vehicle v_i^k),
- a loading point x_i^k (it is 0 if the vehicle uses self-loading),
- an unloading point y_i^k (it is 0 in case of self-unloading),
- amount of the transported commodity w_i^k (can be less than vehicle capacity for commodity u_i^k if location A runs out of commodity u_i^k when loading the vehicle).

The transportation concludes when all commodities have been transported from location A to location B and the vehicles returned to location A. Each transportation cycle P^k

starts and finishes at location A. It consists of the following steps:

1) vehicles wait in queues at the assigned loading points,
2) vehicles are loaded with the assigned commodities,
3) vehicles wait until all of them have been loaded,
4) the convoy travels from location A to location B,
5) vehicles wait in queues at the assigned unloading points,
6) vehicles are unloaded,
7) vehicles wait until all of them have been unloaded,
8) the convoy travels from location B to location A.

Note that the convoy travels from location A to location B with the speed of the slowest loaded vehicle, and from location B to location A with the speed of the slowest unloaded vehicle.

B. Solution Evaluation

The goal is to find such a transportation plan that all commodities are transported in minimal time. The completion time of a transportation plan t is calculated as the sum of the times t^k needed to complete all cycles of the transportation plan:

$$t = \sum_k t^k. \tag{1}$$

Completion time t^k for a transportation cycle is calculated as

$$t^k = t_l^k + t_f^k + t_u^k + t_e^k, \tag{2}$$

where

$$t_l^k = \max \left\{ \max_{j=1,\ldots,n_l} \sum_{\substack{i=1 \\ x_i^k = j}}^{n_v} \frac{w_i^k}{a_{ju_i^k}}, \max_{\substack{i=1,\ldots,n_v \\ x_i^k = 0}} \frac{w_i^k}{a^s_{iu_i^k}} \right\} \tag{3}$$

is the loading time,

$$t_f^k = \max_{\substack{i=1,\ldots,n_v \\ w_i^k > 0}} t_i^f \tag{4}$$

is the travel time of loaded vehicles between locations A and B,

$$t_u^k = \max \left\{ \max_{j=1,\ldots,n_u} \sum_{\substack{i=1 \\ y_i^k = j}}^{n_v} \frac{w_i^k}{b_{ju_i^k}}, \max_{\substack{i=1,\ldots,n_v \\ y_i^k = 0}} \frac{w_i^k}{b^s_{iu_i^k}} \right\} \tag{5}$$

is the unloading time, and

$$t_e^k = \max_{\substack{i=1,\ldots,n_v \\ w_i^k > 0}} t_i^e \tag{6}$$

is the travel time of unloaded vehicles between locations B and A.

A transportation plan P is feasible if each of its cycles P^k satisfies the following constraints:

- Commodity u_i^k assigned to vehicle v_i^k must be available at location A and must be transportable by the vehicle.
- Loading point x_i^k assigned to vehicle v_i^k that transports commodity u_i^k must be able to load commodity u_i^k.
- Unloading point y_i^k assigned to vehicle v_i^k that transports commodity u_i^k must be able to unload commodity u_i^k.
- Self-(un)loading can be assigned to vehicle v_i^k that transports commodity u_i^k only if the vehicle v_i^k can self-(un)load commodity u_i^k.
- The amount of transported commodity w_i^k must be equal to the vehicle capacity unless location A runs out of the assigned commodity while loading the vehicle. Although several vehicles can be loaded with the same commodity when location A runs out of this commodity, all vehicles but one must be fully loaded.

C. Problem Solving

Since the algorithm constructs transportation plans cycle by cycle, the total transportation time cannot be used to evaluate the solutions (the total transportation time is known only after the whole transportation plan has been constructed). Considering only the cycle transportation time would bias the transportation of commodities that are fast to load and unload over the commodities that are slow to load and unload. Therefore, we use a different fitness function capable of evaluating solutions of a single cycle. This function takes into account not only the cycle transportation time (which needs to be minimized), but also the percentages of transported commodity amounts (which should be maximized) and is defined as:

$$f(P^k) = \frac{1}{t^k} \sum_{j=1}^{n_c} \frac{1}{w_j} \sum_{\substack{i=1 \\ u_i^k = j}}^{n_v} w_i^k. \tag{7}$$

To construct a transportation plan, the EA is applied several times—once for each cycle. After optimizing transportation in the first cycle, the EA is run again using the updated commodity amounts at location A thus yielding a solution to the second cycle of the transportation plan. This procedure is repeated until all commodities have been transported from location A to location B.

Here we describe the EA used to find a near-optimal solution to a single transportation cycle. The solutions (chromosomes) are encoded as described in Subsection IV-A, where each quadruple $(u_i^k, x_i^k, y_i^k, w_i^k)$, $i = 1, \ldots, n_v$, is a gene of the chromosome.

Initially, the first population is created and evaluated. S_{pop} chromosomes are created randomly by taking into consideration the constraints described in Subsection IV-A and evaluated using fitness function (7). After the population initialization the chromosomes are evolved for N_{gen} generations. In each generation, the following process is repeated $S_{pop}/2$ times:

1) *Choose two parent chromosomes using tournament selection.*
   Tournament selection, where the tournament size S_{tour} is an algorithm parameter, works as follows. Firstly, S_{tour} chromosomes are drawn randomly from the population. The best of these chromosomes is set to be the first parent. After that, new S_{tour} chromosomes

are randomly chosen from the population and the best among these is set to be the second parent.

2) *Crossover the selected parent chromosomes.*
Multi-point crossover is used, where the number of crossover points N_{cr} and the crossover probability P_{cr} are algorithm parameters. It works as follows. Initially, N_{cr} crossover points are chosen randomly between 1 and $n_v - 1$ and sorted in ascending order. Then, genes are exchanged between each second pair of crossover points.
3) *Mutate each chromosome.*
Genes of a chromosome are mutated with the probability P_{mut}. The mutation changes only one of the first three quantities in the gene by taking into consideration the transportation constraints from Subsection IV-A. If the assigned commodity u_i^k is chosen for mutation, the rest of the gene is repaired to meet the feasibility constraints.
4) *Repair the chromosomes if needed.*
While the variation operators ensure feasibility of genes, the crossover can make the whole chromosome infeasible. Therefore, chromosome repair is often needed. It starts by randomly shuffling the order of the vehicles. The vehicles are then processed in the new order. A vehicle transporting commodity u_i^k is infeasible if previously processed vehicles transported the entire commodity amount or if it does not transport any commodity but there exists a feasible commodity which was not already entirely transported. Such a gene is repaired as follows. Firstly, a randomly chosen feasible and available commodity is assigned to the vehicle. (If there are no feasible and available commodities, the vehicle is not utilized and remains at location A.) Then, the loading and unloading point feasibility is checked and repaired if needed. Finally, the amount of transported commodity is set to meet the feasibility constraints.
5) *Evaluate the chromosomes.*
The chromosomes are evaluated using the fitness function (7).
6) *Update the population.*
Among the two parent and two offspring chromosomes, keep the best two in the population.

The output of the EA is the best solution found.

V. Parameter Tuning

The presented EA operates with static parameter values which have to be specified in advance. Therefore, parameter tuning is needed for obtaining their values. As described in [4] [6], parameter values determine both the quality of solutions found by an EA and the EA performance. Besides, parameters are not always independent [1]. Consequently, a careful selection of parameter values is crucial for the optimization process. Even if an optimal set of parameter values is found for a given problem, it defines the optimal parameter values for solving only the tested problem instances [4]. Therefore, optimal parameter values do not depend on the problem but depend on each problem instance.

Instances of a problem can significantly differ between each other. For example, the presented transportation optimization problem may involve only a few vehicles or even hundreds of vehicles. Solving these two instances would require very different usage of time and space. To increase the EA efficiency, suitable parameter values have to be determined for each problem instance independently. On the other hand, if the number and capacities of vehicles and other problem characteristics do not differ significantly across a set of problem instances, one parameter setting can be used for solving problem instances of this type. Consequently, if there are several types of problem instances, several parameter settings have to be determined. A new problem instance can then be solved using parameter values for the most similar type of problem instances.

Obtaining optimal parameter values for types of problem instances is not trivial. Firstly, problem instance types have to be identified where the crucial step is deciding which problem instances are of the same type. The most straightforward way is to define problem types intuitively, e.g., divide the instances according to their size into small, midsized and large. For example, the number of vehicles in the transportation optimization problem could serve for this purpose. Another possibility would be to consider a larger number of problem characteristics to define instance types. Secondly, good parameter values have to be found for each instance type. The tuned parameter values then have to be checked for differences. If the parameter values for various types of instances do not differ significantly, either the types need to be defined differently or the problem instances probably do not require independent settings of parameter values. The process can be repeated as the number of the encountered problem instances increases.

VI. Experiments

This section presents an example of identifying three types of instances in the transportation optimization problem. EA parameter tuning is then performed for the problem instances together and for each instance individually.

A. Experimental Setup

Three problem instances were generated as representatives of three different types of instances. They comprising four types of commodities, four types of vehicles, two types of loading points and one type of unloading points. These problem instances were aimed as representatives of three types of instances.

Commodities are containers (c_1), large break bulk cargo (c_2), small break bulk cargo (c_3) and dry bulk cargo (c_4).

Vehicles differ regarding their capacity and self-(un)loading capability for each commodity. The first type of vehicles can only transport containers. Such vehicles cannot transport any other kinds of commodities since they do not have sides, and cannot self load or unload since they do not have devices capable of manipulating containers. The second

TABLE I

THE FIRST PROBLEM INSTANCE

		Commodity			
		c_1	c_2	c_3	c_4
Amount		5	2 000	10 000	100 000
Vehicle capacity	v_1-v_2	1	0	0	0
	v_3-v_8	0	40	300	25 000
	v_9	0	0	0	20 000
	v_{10}-v_{13}	0	20	150	0
	v_{15}-v_{20}	0	30	225	0
Vehicle self-loading speed	v_1-v_9	0	0	0	0
	v_{10}-v_{20}	0	12	20	0
Vehicle self-unloading speed	v_1-v_2	0	0	0	0
	v_3-v_9	0	0	0	250 000
	v_{10}-v_{20}	0	12	20	0
Loading point speed	l_1	0	0	0	50 000
	l_2	0	0	0	1 000
	l_3	3	30	90	0
	l_4	4	0	60	0
Unloading point speed	u_1	3	10	30	0
	u_2	0	30	90	0
	u_3	0	20	0	0

type of vehicles can only transport break bulk cargo. Such vehicles are unable to transport containers since they are too small, and are unable to transport dry bulk cargo since they have tilt body. They can have self-loading and unloading devices, e.g., forklift, (un)-loading by hand, etc. The third type of vehicles can transport only dry bulk cargo. Such vehicles are, for example, dump trucks, which are unable to transport containers and break bulk cargo. They cannot self load since they do not have adequate devices, but can self unload by lifting the front part of their bed or by opening special holes at the bottom of it. The fourth type of vehicles can combine the properties of the second and third type of vehicles.

Similarly to vehicles, loading and unloading points differ regarding their (un)loading capabilities for each commodity. The first type of loading points can load only dry bulk cargo (e.g. conveyor belts). The second type of loading points and all the unloading points can load or unload only containers and break bulk cargo (e.g. cranes). There are no unloading points for dry bulk cargo since all vehicles which are able to transport such cargo are also able to self-unload.

The first problem instance consists of transporting large amounts of commodities between two large-scale cargo storages (e.g. between a port and a military base). Example of transported commodities are: containers (c_1), heavy vehicles (c_2), guns (c_3) and grain (c_4). A large number of high-capacity and fast self-(un)loading vehicles, and fast loading and fast unloading points are used for transportation. Moreover, fast and expensive equipment is used since there are no budget limitations (e.g. it is a matter of national security). A detailed problem instance specification is given in Table I.

The second problem instance consists of transporting medium amounts of commodities between two medium-scale cargo storages (e.g. between two storages of a commercial company). Examples of transported commodities are: small containers (c_1), commodities on pallets where each pallet is a single unit (c_2) and sand (c_4). Commodity c_3 is not transported since small cargo is incorporated into pallet cargo (c_2). A medium number of medium-size and average self-(un)loading speed vehicles, average speed loading and unloading points are used for transportation. Equipment of moderate capability is used since the budget of a company is not unlimited. Consequently, highly capable vehicles cannot be used. But on the other hand, a fast transportation is also required, therefore small vehicles cannot be used. Loading and unloading points are chosen analogously. The problem instance is presented in detail in Table II.

TABLE II

THE SECOND PROBLEM INSTANCE

		Commodity			
		c_1	c_2	c_3	c_4
Amount		3	750	0	70 000
Vehicle capacity	v_1	1	0	0	0
	v_2-v_4	0	20	0	0
	v_5-v_8	0	30	0	10 000
Vehicle self-loading speed	v_1	0	0	0	0
	v_2-v_4	0	12	0	0
	v_5-v_8	0	0	0	0
Vehicle self-unloading speed	v_1	0	0	0	0
	v_2-v_4	0	12	0	0
	v_5-v_8	0	0	0	10 000
Loading point speed	l_1	0	0	0	5 000
	l_2	3	30	0	0
	l_3	4	40	0	0
Unloading point speed	u_1	3	10	0	0
	u_2	4	20	0	0

TABLE III

THE THIRD PROBLEM INSTANCE

		Commodity			
		c_1	c_2	c_3	c_4
Amount		0	100	1 000	0
Vehicle capacity	v_1-v_2	0	4	30	0
	v_3-v_5	0	10	60	0
Vehicle self-loading speed	v_1-v_2	0	4	10	0
	v_3-v_5	0	2	5	0
Vehicle self-unloading speed	v_1-v_2	0	4	10	0
	v_3-v_5	0	2	5	0
Loading point speed	l_1	0	10	50	0
	l_2	0	0	40	0
Unloading point speed	u_1	0	10	50	0
	u_2	0	20	0	0

The third problem instance consists of transporting small amounts of commodities from a medium-scale to a small cargo storage (e.g. between regional storage of a commercial company and one of its small shops). Examples of transported commodities are: commodities on pallets where each pallet is a single unit (c_2) and flowers (c_3). Commodities c_1 and c_4 are not transported since a small shop cannot process such commodities: cannot store and does not sell containers or unprocessed dry bulk cargo. A small number of small and slow self-(un)loading speed vehicles, slow loading and unloading points are used for transportation. Such inexpensive equipment is used since the key limitation is the transportation price. Besides low-budget equipment, the minimal transportation time is also required. A detailed overview of the problem instance characteristics is given in Table III.

TABLE IV

TESTED EA PARAMETER VALUES

Parameter	Minimal value	Maximal value	Step size
S_{pop}	50	300	50
P_{mut}	0.00	0.20	0.05
P_{cr}	0.5	1.0	0.1
N_{cr}	1	4	1
S_{tour}	2	$2^{\log_{100} N_{gen}}$	$\times 2$
N_{gen}	100	500	100

TABLE V

NUMBER OF PARAMETER SETTINGS TESTED IN EXPERIMENTS

S_{pop}	Number of values					
	S_{tour}	P_{mut}	P_{cr}	N_{cr}	N_{gen}	Total
50	3	5	6	4	5	1800
100	4	5	6	4	5	2400
150	4	5	6	4	5	2400
200	5	5	6	4	5	3000
250	5	5	6	4	5	3000
300	5	5	6	4	5	3000
						15600

EA parameter values were tuned to obtain the best possible results in terms of the transportation completion time. For the purpose of comparison, this was done for all problem instances together and then for each instance individually. The tested parameter value ranges are presented in Table IV. Consequently, the total number of parameter settings was 15600 (see the explanation in Table V), which were all tested. Since testing one parameter setting takes about 1 minute of CPU time on a 2.66 GHz Intel Xeon processor, we tested each parameter setting on every problem instance 10 times which took about 108 days of CPU time. Average transportation times were recorded for the tested parameter settings and in this way the best settings were determined both over all problem instances and separately for each instance.

B. Results

Since the results covered the entire space of parameter values, the best parameter settings were determined as those resulting in the transportation completion time $t \leq 1.02\, t_{min}$ for each test instance. Then the best parameter settings were processed in two stages. The aim of the first stage was to discover how the best settings differ among problem instances regarding each parameter separately. The results are presented in Figure 1. Each graph shows the distribution of solutions over individual parameter values for all problem instances and for each instance individually.

The aim of the second stage was to analyze the relations among the tuned parameter values and differences in these relations among the problem instances. For this purpose, graphs were constructed for all problem instances cumulatively and for each problem instance individually. Each graph presents the parameter values in parallel coordinates where the parameter values are normalized to take values from the $[0, 1]$ interval. The best parameter settings are drawn as polynomial curves constructed with the Neville's algorithm. This algorithm ensures a unique Lagrange polynomial that

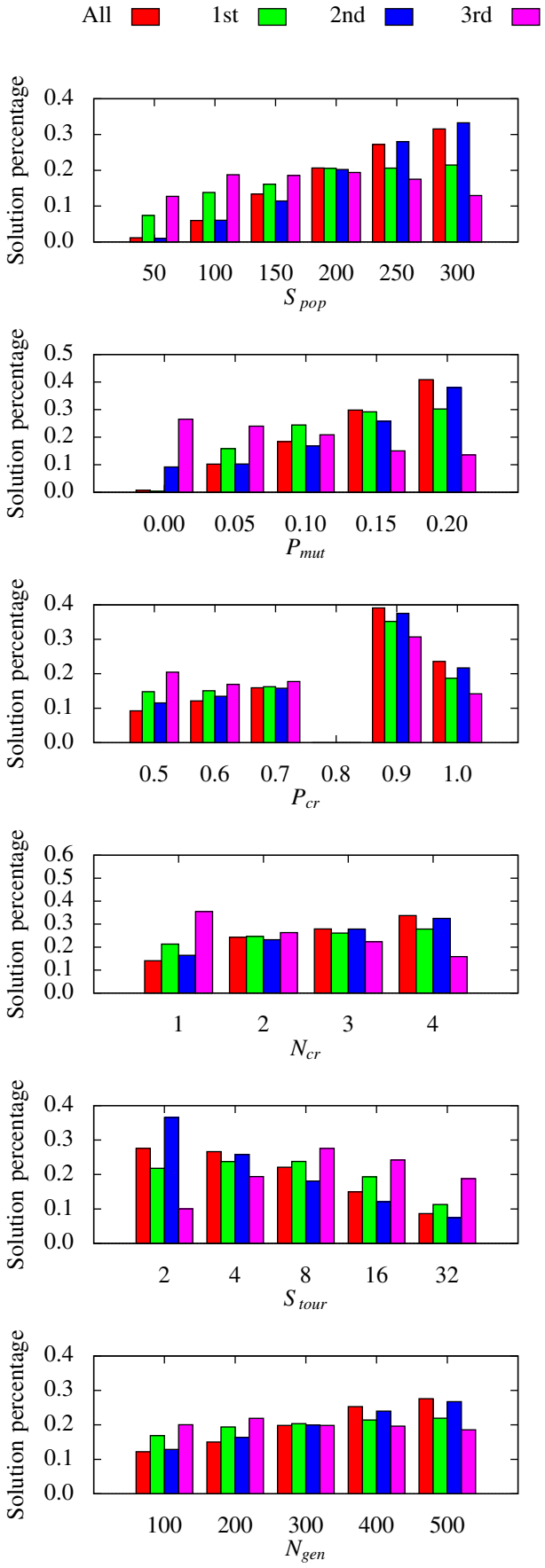

Fig. 1. Distribution of the best solutions over parameter values

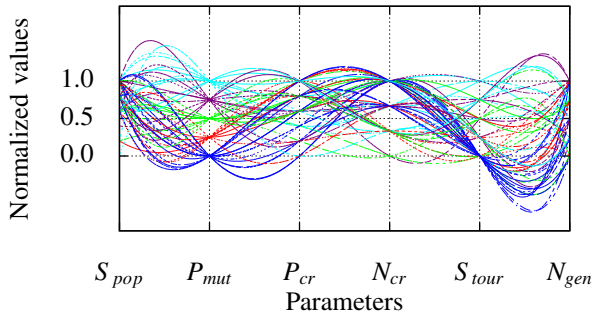

Fig. 2. Distribution of the best parameter values for solving all three problem instances, normalized over the mutation probability

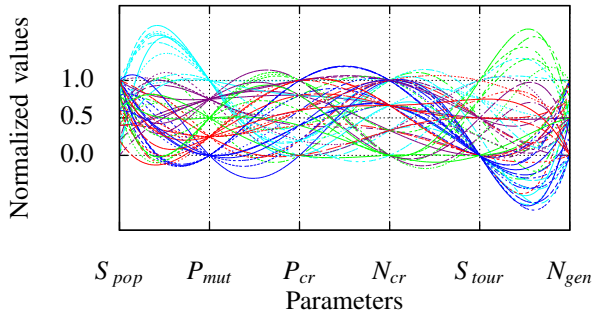

Fig. 3. Distribution of the best parameter values for solving the first problem instance, normalized over the mutation probability

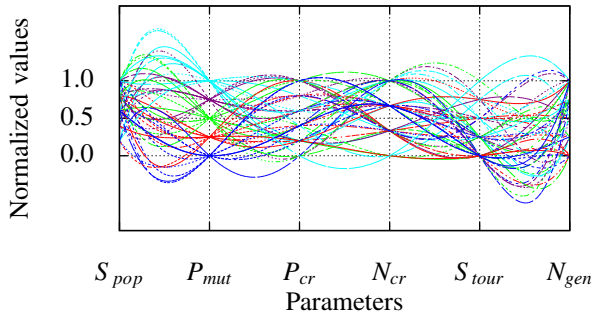

Fig. 4. Distribution of the best parameter values for solving the second problem instance, normalized over the mutation probability

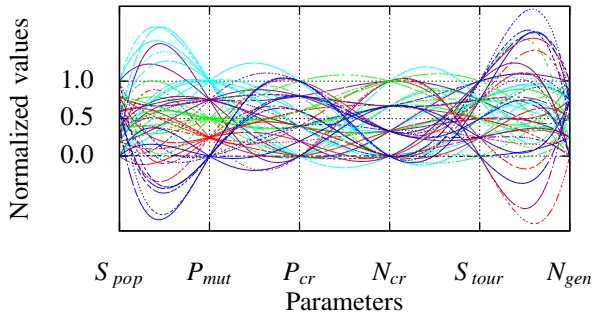

Fig. 5. Distribution of the best parameter values for solving the third problem instance, normalized over the mutation probability

interpolates the data points perfectly [5] [7]. To provide a comprehensible representation, parameter settings for the problem instances were processed separately for each parameter. In the graphs, the polynomial curves are colored denoting different values of a selected parameter. In addition, due to high number of the best parameter settings that are incomprehensible when plotted all together, only 10–20 parameter settings were plotted for each value of a selected parameter. For example, Figure 2 shows the distribution of the best parameter settings over all problem instances with respect to the values of the mutation probability P_{mut}. This analysis resulted in 24 graphs, i.e. 4 operation modes (solving all instances and solving each of the three instances separately) $\times$ 6 parameters. However, because of the space limitation we only show and discuss the graphs for the mutation probability P_{mut} (see Figures 2–5). Other parameter graphs indicate similar relations among parameter values.

VII. Discussion

The main objective of this empirical study was to analyze whether there exist differences among the tuned parameter values for several instances of transportation optimization problem representing specific types of problem instances. Figure 1 shows that a common setting of parameter values for achieving the best results over all problem instances does not exists. Besides, if the parameters are tuned for all problem instances, the achieved values may not be appropriate for specific problem instances. For example, comparing the distribution of the best solutions over population size S_{pop}, the best value for all problem instances and for the second problem instance turns out to be 300, while for the third problem instance it is between 100 and 250. Another significant difference in parameter values can be seen with the distribution over the mutation probability P_{mut}. The best value for all problem instances and the second problem instance is 0.2, while for the first problem instance it is between 0.15 and 0.2, and for the third one it is 0. Other graphs also show notable differences among parameter values for individual problem instances.

An additional objective was to reveal the relations among the tuned parameter values and their variation among problem instances. Figures 2–5 show that relations between the mutation probability and other parameters exist, but only for certain problem instances. For example, the most notable mutation probability value is 0 as shown in Figures 6–9. If all problem instances are taken into consideration and $P_{mut} = 0$, then high number of crossover sites N_{cr} and low tournament size S_{tour} are preferred. The same relation holds for the first problem instance. On the other hand, if the third problem instance is considered, a clear relation between P_{mut}, N_{cr}, and S_{tour} does not exist. These facts indicate the existence of relations among parameter values. While some relations hold for all problem instances, others do only for a subset of problem instances.

In summary, the results show that not only different parameter values are needed to ensure the best EA performance on different types of problem instances, but there also exist

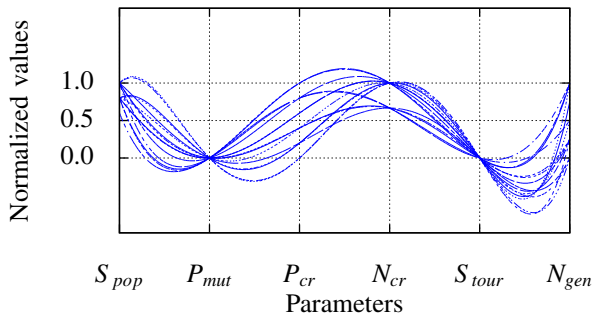

Fig. 6. Distribution of the best parameter values for solving all three problem instances when $P_{mut} = 0$

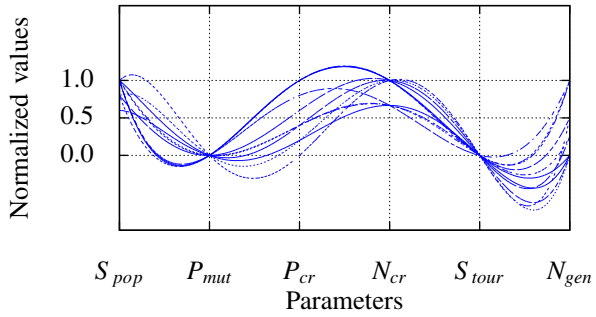

Fig. 7. Distribution of the best parameter values for solving the first problem instance when $P_{mut} = 0$

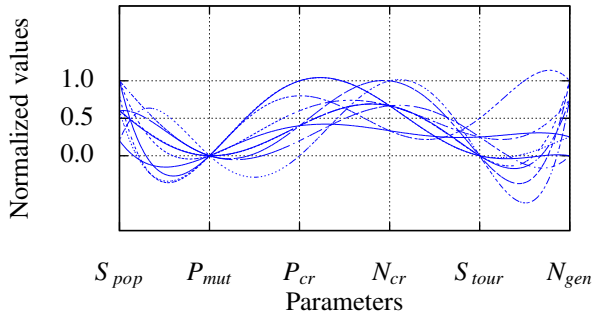

Fig. 8. Distribution of the best parameter values for solving the second problem instance when $P_{mut} = 0$

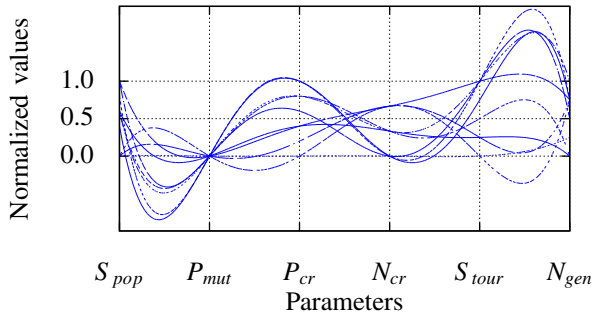

Fig. 9. Distribution of the best parameter values for solving the third problem instance when $P_{mut} = 0$

relations between these parameter values that also vary over types of problem instances. These findings suggest that different parameter values need to be used when dealing with various types of the transportation problem instances.

VIII. Conclusion

Finding good parameter values is important for the efficiency of an EA. In parameter tuning, these values are traditionally determined empirically over a set of problem instances. This work suggests that such a procedure is inappropriate for heterogeneous instances of a problem, since the parameter values tuned on all instances may not perform well on individual instances. A quick conclusion in this situation might be that each problem instance requires its own parameter setting. However, as this is often impracticable, we propose a trade-off approach: identifying types of problem instances and tuning EA parameters for the instance types. New problem instances can then be solved using the tuned parameter values for its type.

The empirical investigation was performed on a limited set of realistic transportation problem instances. In future work, this set will be enlarged by adding several problem instances of each type. Besides, additional tests are planned to prove the appropriateness of determining the problem instance types. Nevertheless, despite the limited test bed, we conclude that parameter tuning for specific types of problem instances improves the EA efficiency in comparison with the algorithm using a single tuned parameter setting on various problem instances.

Acknowledgment

This work was supported in part by the Slovenian Research Agency under research programme P2-0209 *Artificial Intelligence and Intelligent Systems*.

References

[1] P. A. Diaz-Gomez and D. F. Hougen, "Three interconnected parameters for genetic algorithms," *Proc. Genetic and Evolutionary Computation Conference, GECCO 2009*, Montreal, Canada, 2009, pp. 763–770.

[2] A. E. Eiben, R. Hinterding and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[3] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Berlin: Springer, 2003.

[4] A. E. Eiben, Z. Michalewicz, M. Schoenauer and J. E. Smith, "Parameter control in evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms* (Chapter 2), Edited by F. G. Lobo, C. F. Lima and Z. Michalewicz, Berlin: Springer, 2007.

[5] N. Franken, "Visual exploration of algorithm parameter space," *IEEE Congress on Evolutionary Computation, CEC 2009*, Trondheim, Norway, 2009, pp. 389–398.

[6] V. Nannen, S. K. Smit and A. E. Eiben, "Costs and benefits of tuning parameters of evolutionary algorithms," *Proc. 10th International Conference on Parallel Problem Solving from Nature, PPSN X*, Dortmund, Germany, 2008, pp. 528–538.

[7] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*, 2nd edition, Cambridge: University Press, 2002.

[8] S. K. Smit and A. E. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," *IEEE Congress on Evolutionary Computation, CEC 2009*, Trondheim, Norway, 2009, pp. 399–406.