Proceedings of the Federated Conference on Computer Science and Information Systems pp. 57–65
DOI: 10.15439/2018F331
ISSN 2300-5963 ACSIS, Vol. 15

The Practical Use of Problem Encoding Allowing Cheap Fitness Computation of Mutated Individuals

Michal Przewozniczek
Department of Computational Intelligence
Faculty of Computer Science and Management
Wroclaw University of Science and Technology
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
Email: michal.przewozniczek@pwr.edu.pl

Marcin Komarnicki
Department of Computational Intelligence
Faculty of Computer Science and Management
Wroclaw University of Science and Technology
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
Email: marcin.komarcniki@pwr.edu.pl

***Abstract*—The usual assumption in the Evolutionary Computation field is that a cost of computing single fitness function evaluation is at last similar for all cases. Such assumption does not have to be true. In this paper we consider the recently proposed Problem Encoding Allowing Cheap Fitness Computation of Mutated Individuals (PEACh) effect that allows to significantly reduce the computation load of some of the fitness computations that occur during the evolutionary method run. To the best of our knowledge, it is the first experimental analysis that investigates the results of PEACh application to methods solving NP-hard practical problems.**

I. Introduction

THE OPTIMIZATION of computation load consumption by Evolutionary Algorithms (EAs) is a valid and important topic since these are the tools applied to solve hard computational problems. Many techniques were proposed to minimize the expenses for computing the fitness [7], [12], [6]. In this paper, we consider the Problem Encoding Allowing Cheap Fitness Computation of Mutated Individuals (PEACh) effect [17]. PEACh is a recently proposed technique that allows to significantly reduce the computation load spent on a single fitness computation without losing the precision of a result. To the best of our knowledge, except some theoretical experiments proposed in [17], there are no studies that would show the PEACh benefits obtained by applying it to Genetic Algorithms (GAs) solving a hard, practical problem. Therefore, in this paper, we apply PEACh to evolutionary methods applied to solve NP-hard flow optimization problem. The main objective of this paper is to check how significant (if any) is the PEACh influence on results quality and the method efficiency.

The other objective of this paper is to investigate which methods are more suitable to use PEACh benefits. In [17] the standard GA was pointed as a method that is capable of using PEACh optimization only for mutation operator. On the other hand, the multi-population methods employing so-called messy-coding [2], [8], [16], [15] were pointed out as those that should improve their speed more significantly.

This work was supported by the Polish National Science Centre (NCN) under Grant 2015/19/D/ST6/03115

Another important issue that was raised in [17] is the fairness of computation load measurement by using the Fitness Function Evaluation number (FFE). The application of PEACh does not change the method run, except some of the fitness value computations are performed significantly faster. Therefore, in such situations, the use of FFE as a fair computation load measure may be questioned. In this paper, by showing the speed-up, scale we verify if FFE is truly unfair computation load measure for methods using PEACh.

The rest of this paper is organized as follows. In the second section we present the related work, Section 3 contains a definition of the considered practical problem. The fourth section defines PEACh and analyses its possible applications to the considered practical problem. The research results and analysis is presented in Section 5. Finally, the last section summarizes this work and points on most promising future work directions.

II. Related work

In this section, we will present the different propositions of computation load optimization techniques employed in the Evolutionary Computation field. We will also discuss some related issues, eg. the fairness of the computation load measurement with the use of Fitness Function Evaluation number (FFE). Finally, we will briefly present the benefits of using multi-population approaches with a dynamically changed number of subpopulations since one of the methods considered in this paper is employing such techniques.

In some of the papers concerning Evolutionary Algorithms (EAs) applied to solve practical problems it is pointed that the computation load necessary to compute particular individual's fitness may be significantly decreased if the fitness value of similar (but not necessarily the same) individual is known. In [9], [10] different Resource-Constrained Scheduling Problem (RCSP) version are considered. To compute fitness of any individual, the problem solution that is represented by this individual must be constructed first. Then, the constructed solution is rated, and the fitness is computed. Some of the presented methods introduce small changes to already known and already rated individuals. Thus, the fitness of new individual that is a result of small genotype modification may be

computed in two different ways. It may be computed in a usual way by constructing and rating the solution. However, it is also possible to copy the solution of an old, already rated individual, modify it and rate it. In RCSP problem the computation load necessary for solution construction is significantly higher than the computation load required for rating the constructed solution. Thus, in NEH2 heuristic (Nawaz, Enscore, Ham) proposed in [9] the fitness of new solutions is computed by modifying the already known solutions rather than by building new solutions from scratch. In result, NEH2 can use a higher FFE number than its predecessor – NEH heuristic.

Surrogate model [6] is an interesting technique of computation load usage optimization that is useful when single fitness evaluation takes significantly long time (e.g. minutes, hours, or even days). In such situation, the use of evolutionary methods may be limited. The idea is to propose a new problem model that would mimic the real model but would be significantly cheaper to evaluate. Therefore, the use of surrogate model enables the use of EAs for problems to which they would be otherwise inapplicable due to practical reasons. The surrogate model is similar to PEACh effect considered in this paper because it optimizes the cost of fitness value computation. However, the difference is that PEACh leads to an exact fitness value, while surrogate model offers only the approximated fitness value.

Another technique that allows decreasing the computation costs is fitness caching [7], [12]. Its idea is based on storing the information about fitness values computed for the particular genotypes. When such knowledge is available instead of computing fitness the method checks if the fitness for the particular genotype was not already computed. If so, then instead of computing fitness the stored fitness value is returned. Such technique may significantly decrease FFE. However, the drawback is that checking if the fitness for the particular genotype was not computed before becomes more and more expensive in time as the list gets longer. At some point, the benefits brought by omitting the fitness computation may be exceeded by costs generated by the list search. Another issue is that the list of already rated genotypes may consume high amounts of memory. Therefore in [7] two different fitness caching techniques are described - *brutal fitness caching* (that is the technique described above) and *population fitness caching*. The population fitness caching works in the same way as its brutal version but instead using the genotypes list, the optimization is limited to the search through the current population. The research presented in [7] proposes an analysis of benefits brought by both fitness caching techniques. The research is based on modern evolutionary methods: Linkage Learning Genetic Algorithm (LTGA) [18], Dependency Structure Matrix Genetic Algorithm II (DSMGA-II) [5] and Parameter-less Population Pyramid (P3) [3]. Another important issue is shown in [7] is that when any fitness caching is used the FFE is not a reliable computation load measure when a method gets stuck. The reason for this situation is that once a method gets stuck it simply loses the capability of proposing new solutions, which have not been investigated yet. If so, then at some point all, or almost all fitness computation requests are cached. In the research presented in [7] FFE per iteration may drop to zero for the whole remaining method run. Thus, a different computation load measure than FFE number is necessary because in the described situation FFE only shows that method is not consuming any computation load at all which is not true.

The issue of fair computation load measurement is also addressed in [14]. One of the assumptions of using FFE as a fair computation load measure is that the computation load required to compute a single fitness value is the same or, at least, similar. The research presented in [14] show that this assumption is not always true. For instance, the computation load may be dependent on the genotype. Another requirement to use FFE as a fair computation load measure is that the dependency between the overall computation load used by a method and FFE should be close to linear. In other words, the computation load necessary to compute the fitness value is significantly higher than the computation load used for all other method activities. If this condition is not true, then the use of FFE as computation load measure may not be reliable. This issue is discussed in details in [8] on the base of Bayesian Optimization Algorithm (BOA). During its run, at each iteration BOA constructs the model of gene dependencies. When the genotype is long the computation load necessary for model construction significantly exceeds the computation load spent on all other method activities making it ineffective.

III. Flow assignment in computer networks

The problem of flow assignment in computer networks is one of the main problems in the field of network design [11]. The other are the capacity assignment, flow and capacity assignment, topology, flow, and capacity assignment. In the flow assignment problem, the solution shall satisfy the set of demands. Each demand defines an amount of information that is to be sent between a particular pair of network nodes. The list of demands as well as the network topology are given and cannot be modified. For each demand, a route in the network must be set to satisfy it. Thus, the solution to the problem is a list of routes (one route for one demand) that satisfy all demands and do not break the network links capacity constraint. The solution quality may be measured in many different ways. Here, we employ the Lost Flow in Link (LFL) function. LFL describes how well the network topology is prepared for the link breakdown scenario. The optimization of LFL value increases the network survivability and the quality of service. The problem denominated as WP_LFL [15] and is NP-complete [11].

The notation used to represent the WP_LFL problem is presented below.

Sets

V - set of n vertices representing the network nodes
A - set of m arcs representing network directed links
P - set of q connections in the network

$\prod_p$ - the index set of candidate working paths (routes) for connection p
X_r - set (selection) of variables x_k^p, which are equal to one. X_r determines the unique set of currently selected working paths
Indices
p - connections (demands) in the network, used as subscript
k - candidate routes, used as superscript
a - arcs (directed links), used as subscript
r - selections, used as subscript
Other
$o(a)$ - the start node of arc a
$d(a)$ - the end node of arc a
Constants
δ_{pa}^k - equal to1, if arc a belongs to path k realizing connection p; 0 otherwise
Q_p - volume (estimated bandwidth requirement) of connection p
c_a - capacity of arc a
Variables
x_p^k - decision variable, which is 1 if working route $k \in \prod_p$is selected for connection p and 0 otherwise
f_a - flow of arc a
$g_v^{in} = \sum_{i:d(i)=v} f_i$ - aggregate flow of incoming arcs of v;
$e_v^{in} = \sum_{i:d(i)=v} c_i$ - aggregate capacity of incoming arcs of v;
$g_v^{out} = \sum_{i:o(i)=v} f_i$ - aggregate flow of outgoing arcs of v;
$e_v^{out} = \sum_{i:o(i)=v} c_i$ - aggregate capacity of outgoing arcs of v;

The $o : A \to V$ and $d : A \to V$ functions denote the origin and destination node of each arc. For each $a \in A$ the set of incoming arcs of $d(a)$ except a $in(a) = \{k \in A | d(k) = d(a), k \neq a\}$, and the set of outgoing arcs of $o(a)$ except a $out(a) = \{k \in A | o(k) = o(a), k \neq a\}$ are defined.

Definition 1. *The global non-bifurcated m.c. flow denoted by $\underline{f} = [f_1, f_2, ..., f_m]$ is defined as a vector of flows in all arcs. The flow $\underline{f}$ is feasible if for every arc $a \in A$ the following inequality holds*

$$\forall a \in A : f_a \leq c_a \tag{1}$$

Inequality 1 ensures that in every arc, flow is not greater than capacity. This inequality is called the capacity constraint.

For the sake of simplicity, the following function is introduced

$$\epsilon(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > k \end{cases} \tag{2}$$

The analysis of local repair properties is based on a scenario where the failure of arc $k \in A$ is considered. If local repair is used then flow on the arc k must be rerouted by the origin node of k. Therefore, residual capacity of arcs outgoing from node $o(k)$ except arc k is a potential bottleneck of the restoration process. Since

$$f_k \leq \sum_{i \in out(k)} (c_i - f_i) \tag{3}$$

then the flow of the failed k can be restored using the residual capacity of other links leaving the origin node of k. Otherwise, if

$$f_k > \sum_{i \in out(k)} (c_i - f_i) \tag{4}$$

then some flow of the failed link k cannot be restored because the residual capacity of other arcs leaving the origin node of k is too small. As a consequence, the 100% restoration is not possible and some flow of k is lost. Applying formulas 3 and 4, and the $g_{o(k)}^{out}$, $e_{o(k)}^{out}$ definitions, the LA^{out} function is defined as follows.

$$LA_k^{out}(\underline{f}) = \epsilon(g_{o(k)}^{out} - (e_{o(k)}^{out} - c_k)) \tag{5}$$

Formula 5 defines the flow lost for arc k as dependent on the whole flow leaving the origin node of k. Therefore, the function of flow lost for all arcs leaving node v is defined as follows.

$$LN_v^{out}(\underline{f}) = \sum_{a:o(a)=v} \epsilon(g_v^{out} - (e_v^{out} - c_a)) = \sum_{a:o(a)=v} LA_v^{out}(\underline{f}) \tag{6}$$

Function $LN_v^{in}(\underline{f})$ is analogous to $LN_v^{out}(\underline{f})$ and defines how much flow is lost in the arcs incoming to node v. The goal of defining a function that measures the network preparation to link breakdown is realized in 7.

$$LFL(\underline{f}) = \sum_{v \in V} (LN_v^{in}(\underline{f}) + LN_v^{out}(\underline{f}))/2 \tag{7}$$

More details about the LFL may be found in [13]. The optimization problem, WP_LFL [13], [15] is defined as follows.

$$\min_{\underline{f}} \quad LFL(\underline{f}) \tag{8}$$

subject to

$$\sum_{k \in \prod_p} x_k^p = 1 \quad \forall p \in P \tag{9}$$

$$x_k^p \in \{0, 1\} \quad \forall p \in P, \forall k \in \prod_p \tag{10}$$

$$f_a = \sum_{p \in P} \sum_{k \in \prod_p} \delta_{pa}^k x_p^k Q_p \quad \forall a \in A \tag{11}$$

$$f_a \leq c_a \quad \forall a \in A \tag{12}$$

Condition (9) guarantees that the each connection can use only one working route. Constraint (10) ensures that decision variables are binary ones. Formula (11) defines a link flow. Finally, (12) denotes the link capacity constraint. The

WP_LFL problem given by (8)-(12) is a 0/1 NP problem with linear constraints. For this problem, the solution space that includes all possible paths for each connection is large even for relatively small networks. Therefore, the problem is considered hard. Let us consider 2500 demands and ten routes available for each demand. The number of solutions (not all may be feasible) 10^{2500}.

IV. PEACh effect in considered practical problem

In this section, we present PEACh idea proposed in [17]. Since PEACh benefits are strictly dependent on the problem encoding, in the second subsection we give a detailed description of solution encoding for the considered WP_LFL problem. Finally, in the last subsection, we present the two considered competing methods and discuss whether they are suitable to use PEACh benefits.

A. General PEACh description

Let us consider a situation in which fitness computation process is built from two stages - solution creation and solution rating. For some problems, the main computational cost is paid on the solution creation stage, while the process of rating the created solution is relatively cheap [17]. In such situation, if we wish to rate an individual but we know a similar solution that was already rated it is reasonable to copy the rated solution, modify it and rate the resulting solution.

Let us consider the Traveling Salesman Problem (TSP). TSP can be represented as a graph $G = (V, E)$, where V is a set of n vertices (cities) and $E = \{ei, j\}_{n\times n}$ is a set of edges that represent connections between cities. The distance function $d : E \to \Re^+$ is used to assign each edge e a distance value. The goal is to find a Hamiltonian cycle of minimal distance that visits each vertex only once. Let us assume that in the considered TSP instance we need to visit 1000 cities, the genotype encodes the solution by storing a list of genes that are city identifiers. Some individual that was already rated is being mutated by changing two genes (cities order) in its genotype. To compute fitness for the mutated version of the individual, we may compute the cost generated by using 1000 routes that are encoded by the genotype of the mutated individual. Another option is to copy the fitness of individual before mutation, subtract from it the distance of two routes that were removed and add the distance yield by two new routes. Note, that the second option requires significantly lower computation load.

We may state that PEACh benefits are used if the following condition holds

$$cost(X) \gg cost(X_{mut}, fitInfo(X)) \tag{13}$$

subject to

$$diffr(X, X_{mut}) \ll size(X) \tag{14}$$

where
X, X_{mut} - individuals genotypes
$cost(X)$ - the computation load that must be paid to compute the fitness of X without any prior knowledge
$fitInfo(X)$ - the additional information about data produced during X fitness value calculation process (including X fitness value if necessary)
$cost(X, addInfo)$ - the computation load necessary to compute the fitness of X with additional information taken from the other fitness value calculation operation
$diffr(X, Y)$ - the number of genotype positions for which genotype X is different than the genotype Y
$size(X)$ - the number of genotype positions in genotype X

If an additional information (from another fitness value computation operation) is available, then the computation load necessary for the fitness value computation will be significantly lower than if the same computation was done without the additional knowledge (inequality (13) is true). Condition (14) guarantees that the additional information is supported by the fitness calculation process for similar genotype.

The above definition proposed in [17] does not define a unit of computation load amount (returned by $cost(X, addInfo)$ and $cost(X)$). In this paper, similar as in [17], we use the computation time. Another available choice is the number of processor instructions. Note, that FFE is not an allowed choice, since the amount of computation load necessary to compute a single fitness evaluation may significantly differ.

B. PEACh in WP_LFL

In WP_LFL the network topology is given. The network links connecting the nodes are directed, so the particular link transfers data only in one direction. Therefore, the network may be represented as a directed graph. The list of demands defining the nodes pairs between the communication channels must be established is also entry information. Each demand except the start and destination node defines the volume of the demanded communication channel. Each connection is using only one route. Thus, the solution to the problem is a set of routes (each route proposed for one demand). Many communication channels may go through a single network link. The summarized volume going through a single network link may not exceed the link capacity. The example of the 4-demand problem solution encoded in the GA-like manner may be as follows: [(4) (6) (11) (13)]. In this solution the first demand will use route number 1, second demand will use the sixth route, etc. Note, that the solution does not have to be feasible, i.e. the link capacity constraint may be broken.

In Fig. 1 we present the example of the network state in the form of the two-dimensional matrix. The numbers in the matrix represent the available capacity of network links. For instance, the link from node A to node B has 24 capacity units left. For the network state presented in the upper part of the figure, we wish to set up the connection channel using the route from node A through nodes B and C to node D. The demand size (volume) of the connection channel is 4 capacity units. In the result of this operation, the available capacity in links A to B, B to C and C to D is decreased by 4. In the WP_LFL instances consider in the research present in this

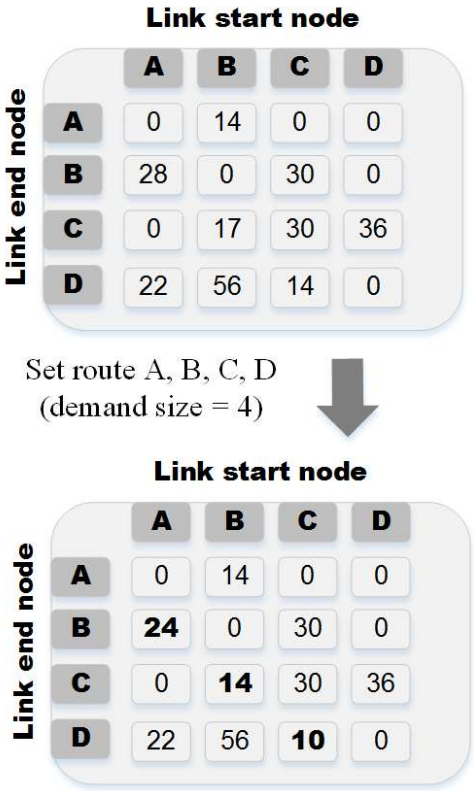

Fig. 1. The example of route establish

paper we use 1260 up to 2500 demands. To compute fitness of any individual we have to create a solution (by setting all connection channels for all demands) and then rate it (compute LFL function value for all links and summarize it). For the considered test cases the time necessary to construct the problem solution (T_{model}) is significantly larger than the time necessary to rate the constructed solution ($T_{quality}$). Therefore, the decreasing T_{model} time, shall significantly decrease the overall time necessary for fitness computation. Let us consider the following situation. The individual X encoding a solution to 2500-demands WP_LFL problem instance was rated and cost of this operation was $cost(X) = T_{model} + T_{quality}$. Then, individual X was mutated (one of its genes was modified) creating individual X_{mut}. We can compute fitness of individual X_{mut} without using any additional knowledge. If so, then $cost(X_{mut}) \approx costX$. However, we can also compute the fitness of X_{mut} differently and use the model created for individual X. It is enough to unset one connection channel that was encoded by individual X (set this channel with negative volume) and set the new channel that was defined in individual X_{mut}. In such situation we need to perform two channel setting operations instead of 2500 route setting operations $cost(X_{mut}, fitInfo(X)) = 2/2500 \cdot T_{model} + T_{quality}$. Since for the considered test cases $T_{model} \gg T_{quality}$, it is true that $cost(X) \gg cost(X_{mut}, fitInfo(X))$. Thus, we may expect that using the PEACh effect may significantly decrease the computation load consumed by a method during its run.

C. Competing methods and PEACh suitability

In this section, we present the two methods chosen for tests. The first is Hierarchical Evolutionary Algorithm for Flow Assignment in Non-bifurcated Commodity Flow (HEFAN 2.2) [13]. HEFAN 2.2 is a standard GA that is dedicated to solving flow optimization problem. The individuals are encoded in a standard GA-manner presented in the previous section. Since considering all available routes between every nodes pair would lead to the combinatorial explosion and would make the problem intractable, at the beginning of the run HEFAN 2.2 uses four shortest routes between each node pair and all routes that are not longer than 1-, 2-, 3-, and 4-hops. Such route set may be not enough to construct some high-quality solutions. Therefore, HEFAN 2.2 uses two problem-dedicated operators that were designed to propose new routes, namely the low-level crossover and low-level mutation operators. The idea behind both low-level operators is to interpret a single route (that is an ordered list of nodes) as an individual and apply to it standard GA operators. The routes resulting from low-level operators may not be feasible in the particular network topology. Therefore, the repair procedure is applied to their results. The details about dedicated operators employed by HEFAN 2.2 may be found in [13].

The second considered method is Multi Population Pattern Searching Algorithm for Flow Assignment in Non-bifurcated Commodity Flow (MuPPetS-FuN) [15], based on MuPPetS [8]. In MuPPetS-FuN two types of individual are employed. The classical GA-like individuals called *Competitive Templates* (CT) are used together with messy-coded individuals [2] called *viruses*. The messy-coded individuals do not encode the complete problem solution, but only a part of it. To rate such individual, the missing genes must be first supplemented. Therefore, each CT (complete problem solution encoded in GA-like manner) has a crowd of viruses assigned to it. The genes that are missing in each virus are supplemented from CT the virus is assigned to. Each virus population is separate, and the number of their populations is equal to the number of CTs. The virus population evolution process may be found as the optimization of the CT the viruses are assigned to - after their evolution is finished, if the fitness of best-found virus is better than its parental CT, then the genes from the virus infect the CT, improving it. The important feature of messy-coding concerning PEACh effect is that viruses encode only a small part of a complete problem solution. Thus, each virus may be interpreted as a modification of its parent CT. Since, $l_{vir} \ll size(CT)$, where l_{vir} is the genotype length of a virus, then the process of virus fitness computation seems to be suitable for using the benefits of PEACh effect.

V. Results

In this section, we present the results of experiments performed for two methods dedicated to solving the considered WP_LFL problem. HEFAN 2.2 is based on the idea of standard GA, while MuPPetS-FuN Active is a multi-population

method that uses messy-coding and dynamically manages the number of maintained subpopulations (i.e., during the method run new subpopulations are created, and some subpopulations are deleted). Since HEFAN 2.2 is based on standard GA idea, the use of PEACh effect will be limited only to mutation operator. On the other hand, MuPPetS-FuN uses messy-coding which is suitable for gaining the benefits brought by PEACh. The objective of the research is twofold. First, we wish to experimentally check how significant may be the optimization of computation load usage when PEACh is employed and how this optimization is dependent on the method type. Second, we wish to check how significantly the use of PEACh may influence the results quality for the considered WP_LFL problem.

The rest of this section is organized as follows. In the first subsection, we report the experiment setup. In the second subsection, we show and comment the differences in the computation load usage optimization brought by PEACh. The third subsection presents the influence of PEACh effect on the results quality for the two considered methods types. Finally, the last subsection contains the results discussion.

A. Experiment setup

In the experiments, the time-based stop condition was used. As pointed out in sections II and IV, since PEACh is employed, the use of FFE as a stop condition is doubtful for the research presented in this paper. The experiments were executed on PowerEdge R430 Dell Server Intel Xeon E5-2670 2.3 GHz 64GB RAM with Windows 2012 Server 64-bit installed. To ensure that the computation load used in each experiment is equal, the number of computation processes was always one less than a number of available CPU nodes. The time limit was set to 3 hours. All methods were programmed in C++, share all the possible pieces of code and are single-threaded. The experiments are executed in the clean environment – i.e., no other resource consuming processes are running, and the number of executed experiments is always one less than the number of available processor cores (1 core is spared for the operating system activities). Such assumptions are the same as in [7], [15], [8] and shall allow for fair comparison.

To compare the performance chosen methods we use ranking, defined as follows. The best method for a particular experiment receives the number of points equal to the number of competing methods; the second method receives one point less, etc. If more than one method takes the same place, then all such methods receive the same number of points as for one method. For instance, there are three competing methods A, B and C. Methods A and B were the best, and receive 3 points. Method C was the worst one – it receives 1 point.

The considered experiments may be grouped concerning the network type or flow parameters. Six different network topologies were considered. The networks parameters (minimal, maximal and average node degree) might be found as typical [19]. The mesh of five networks is irregular, the mesh of one network is grid-like. The parameters of all considered networks are presented in Table I.

TABLE I
CONSIDERED NETWORKS PARAMETERS

Network	**104**	**114**	**128**	**144**	**162**	**Grid**
Node number	36	36	36	36	36	36
Arc number	104	114	128	144	162	120
Minimal node deg.	2	2	3	3	3	2
Maximal node deg.	5	5	6	6	6	4
Average node deg.	2.89	3.17	3.56	4	4.5	3.33
Topology	Irregular mesh					Regular mesh

TABLE II
EXPERIMENT GROUPS PARAMETERS

Experiment group	**Group A**	**Group B**	**Group C**
Arc capacity	4800	4800	$km \cdot 1200$,where km=1,...,8
Connections to set	1260	2500	2500
Connection choice	1 for each pair	random	random
Demand size	equal for all connections	random	random

The classification using the demanded flow parameters divides the experiments into three groups: A, B and C. Each experiment group is characterized by arc capacities used, a number of connections to be set (demands) and the way the connections were chosen. The OC-12 and OC-48 standards, typical for transportation networks [4], were taken into consideration at the arc capacity design. The parameters for each experiment group are presented in Table II. The units of demands sizes and arc capacities were abandoned which is frequent for the papers concerning the problems of flow assignment in computer networks [11], [13], [15].

Ten experiments were used for each network and experiment group. Thus, the total number of test cases was $6 \cdot 3 \cdot 10 = 180$. The considered problem is NP-complete [4]. As presented in table II the number of demands is 1260 or 2500. In the employed solution encoding each gene refers to a single demand, which makes gene number equal to demand number. If for each demand we consider 10 different routes, then the number of solutions that may be encoded is equal to 10^{1260} or 10^{2500}.

The methods settings were adopted from [15] and are presented in Tables III and IV.

B. PEACh benefits depending on method type

One of the main objectives of this paper is to check how significant is the influence of PEACh effect on computation load optimization for the considered problem depending on

TABLE III
MUPPETS-FUN ACTIVE SETTINGS

Parameter name	**MuPPetS-FuN Active**
Virus generations	100
Virus subpopulation size	100
Cut	0.21
Splice	0.30
Mutation	0.20
Low Level Crossover	0.60
Low Level Mutation	0.20

TABLE IV
HEFAN 2.2 SETTINGS

Parameter name	**HEFAN 2.2**
Population size	1000
Crossover	0.9
Mutation	0
Low level crossover	0.1
Low level mutation	0.4
Uniform crossover	0.5
Crossover	0.9

TABLE V
FFE INCREASE RATIO CAUSED BY PEACH EFFECT IN MUPPETS-FUN ACTIVE RUNS

	Avr	**St. dev.**	**Min**	**Max**	**Mean**
All	207.92%	48.20%	123.14%	311.01%	207.18%
104	241.78%	34.17%	188.03%	311.01%	239.90%
114	216.76%	34.42%	170.45%	279.07%	207.64%
128	212.07%	23.90%	171.90%	265.80%	211.09%
144	158.89%	22.02%	129.44%	197.67%	157.62%
162	149.49%	20.17%	123.14%	187.52%	143.62%
Grid	255.66%	21.37%	225.07%	291.69%	261.73%
Group A	174.72%	39.80%	123.14%	251.10%	178.37%
Group B	192.74%	40.34%	140.48%	276.49%	205.40%
Group C	220.23%	45.64%	144.99%	311.01%	213.23%

the method type. Therefore, we compare the amount of FFE done by MuPPetS-FuN Active and HEFAN 2.2 when using and not using the benefits of PEACh effect. To assure that the comparison is precise, each experiment was executed with the same random seed. The runs in which the optimal solution was found were excluded from the comparison. The FFE increase ratio (FFE for the run using PEACh effect is divided by FFE for the run without PEACh) is presented in Tables V and VI.

The results presented in Tables V and VI show that using PEACh effect is far more beneficial for MuPPetS-FuN Active. For MuPPetS-FuN, the minimum increase obtained in all 180 runs is over 123%, while the maximum ratio for HEFAN 2.2 is less than 112%. Such results are expected since during most of its run MuPPetS-FuN process messy-coded individuals, which genotypes are much shorter than the genotype of full GA-like coded problem solution. Thus, PEACh effect benefits may be used at every fitness computation of messy-coded individual. On the other hand, for HEFAN 2.2 whish is based on a standard GA, PEACh is only beneficial during mutation.

It is also interesting that for MuPPetS-FuN Active PEACh

TABLE VI
FFE INCREASE RATIO CAUSED BY PEACH EFFECT IN HEFAN 2.2 RUNS

	Avr	**St. dev.**	**Min**	**Max**	**Mean**
All	107.98%	1.98%	104.14%	111.95%	108.14%
104	109.08%	2.24%	104.21%	111.95%	109.98%
114	107.91%	1.94%	104.64%	111.05%	108.04%
128	107.42%	2.37%	104.14%	111.76%	106.24%
144	107.71%	1.58%	105.17%	110.41%	108.48%
162	107.13%	0.94%	105.50%	108.79%	107.18%
Grid	107.98%	1.84%	104.43%	110.69%	108.56%
Group A	105.93%	1.06%	104.21%	108.42%	105.78%
Group B	109.12%	1.93%	104.14%	111.76%	109.14%
Group C	108.65%	1.45%	105.27%	111.95%	108.65%

TABLE VII
THE INFLUENCE OF PEACH EFFECT ON METHOD EFFECTIVENESS (RANKING)

		PEACh		**No PEACh**	
	LRH	**MuPPetS**	**HEFAN**	**MuPPetS**	**HEFAN**
All	2.15	**2.98**	2.33	2.62	2.12
104	2.43	**3.40**	2.97	2.70	2.77
114	1.77	**3.00**	2.03	2.60	1.87
128	1.87	**3.03**	2.10	2.43	1.80
144	1.43	**2.90**	2.50	**2.90**	2.17
162	1.80	2.43	2.37	**2.53**	2.13
Grid	**3.60**	3.13	2.03	2.53	1.97
Group A	1.12	**2.40**	2.34	2.30	2.38
Group B	1.58	2.50	**2.58**	2.02	2.36
Group C	2.88	**3.96**	2.26	3.58	1.70

effect is significantly more beneficial for some test case groups. For instance, the highest FFE increase ratio is found for networks *Grid* and *104*. On the other hand, FFE increase ratio is the lowest for experiments using networks *144* and *162*. The detailed analysis of this phenomenon is out of this paper scope and is one of the interesting future work directions. The reasonable explanation seems to be that in the experiments for networks *144* and *162* the average length of messy-individual genotypes is significantly longer than in the experiments for *Grid* and *104* networks. The longer is the messy-coded individual's genotype, the less significant is the fitness function computation speed-up caused by PEACh effect. This observation seems to be supported by the FFE ratio observed experiments in groups A, B, and C. The full GA-like encoded solution in experiments from group A contains 1260 genes, while in experiments in groups B and C this is 2500 genes. We may expect that in experiments of from group A, the average genotype of messy-individual contains a larger percentage of all necessary genes than in two other groups.

C. PEACh influence on results quality

In this section, we compare the effectiveness of both considered methods, depending on PEACh effect. In these experiments the randomizer seed was random. Thus, it is possible that a method version with PEACh may return a lower quality result than a version without PEACh. Note, that it was impossible for the experiments considered in the previous subsection because when the seed is set manually both method versions (with and without) were performing the same run. The only difference was that version with PEACh was working faster, so it was able to perform a higher number of iterations. The comparison of results quality in Table VII. In the experiments presented in this subsection the MuPPetS-FuN and HEFAN 2.2 are also compared with Lagrangian Relaxation Heuristic (LRH) [13], [15]. LRH is a hybrid algorithm that joins the Flow Deviation for Primary Routes algorithm [1] and Lagrangian Relaxation. It uses sub-gradient optimization to determine Lagrangian coefficients and was shown effective [13] in solving flow assignment problems.

As presented in Table VII the methods employing PEACh effect are the most effective for all experiment groups except two. The first is the Grid network group for which the most

TABLE VIII
THE COMPARISON OF MUPPETS-FUN EFFECTIVENESS WITH AND WITHOUT PEACH EFFECT ON THE BASE OF p-VALUE REPORTED BY SIGN TEST

	with PEACh better or equal	Equal	with PEACh worse or equal
All	**100.00%**	0.00%	0.00%
104	**99.95%**	0.37%	0.19%
114	**98.94%**	7.68%	3.84%
128	**100.00%**	0.07%	0.04%
144	59.27%	**100.00%**	59.27%
162	50.00%	**100.00%**	68.55%
Grid	**99.88%**	0.94%	0.47%
Group A	**92.52%**	28.10%	14.05%
Group B	**99.99%**	0.06%	0.03%
Group C	**100.00%**	0.01%	0.00%

TABLE IX
THE COMPARISON OF HEFAN 2.2 EFFECTIVENESS WITH AND WITHOUT PEACH EFFECT ON THE BASE OF p-VALUE REPORTED BY SIGN TEST

	with PEACh better or equal	Equal	with PEACh worse or equal
All	**99.98%**	0.07%	0.04%
104	**96.08%**	18.92%	9.46%
114	**92.83%**	33.23%	16.62%
128	**98.94%**	7.68%	3.84%
144	**96.82%**	16.71%	8.35%
162	**98.46%**	9.63%	4.81%
Grid	73.83%	**83.18%**	41.59%
Group A	50.00%	**100.00%**	64.94%
Group B	**99.00%**	4.70%	2.35%
Group C	**99.98%**	0.09%	0.04%

effective is LRH, the second is 162 network for which the most effective MuPPetS-FuN Active without PEACh effect. In the experiments employing the Grid network, LRH simply seems to be a more suitable method. Such observation is consistent with the previous research in this area [15]. For experiments using 162 network, the explanation may be as follows. MuPPetS-FuN Active seems to be effective in solving test cases from this group. The PEACh influence on computation load used by MuPPetS-FuN Active was the lowest for these experiments (less than 150% of average ratio presented in Table V), so it is likely that such result is the effect of noise. To check if the benefits of PEACh effect are statistically significant we have used Sign Test. The results are presented in Table VIII.

As presented in Table VIII the use of PEACh effect is not statistically significant for the method effectiveness for experiments using network *144* and *162*. Such results are not surprising since for both of these subgroups the average FFE increase ration was the lowest. The third subgroup for which the influence of PEACh benefits on results quality does not seem statistically significant are experiments from Group A. Note, that for this subgroup average FFE increase ratio was the third lowest one.

The same statistical tests were performed for HEFAN 2.2. As expected, for this method the results are less convincing - the p-value of tests checking that HEFAN 2.2 effectiveness with and without PEACh is equal is significantly higher and is above 9% for 6 of 10 experiment groups. Nevertheless, when all experiments are taken into account, the results are decisive. Thus, it is allowed to state that for HEFAN 2.2 the influence of PEACh significantly affects the results quality.

D. Results discussion

The results presented in this paper show that the influence of PEACh effect may significantly optimize the computation load used by a method. Thus, it may influence the method effectiveness when the available resources are limited. The research presented in this paper show that some methods are more suitable to use PEACh benefits than other. Here, MuPPetS-FuN Active that employs messy-coding was able to perform from 123% up 311% of FFE that would be computed without using PEACh. Such change seems significant. Note, that messy-coding is not the only mechanism suitable for using PEACh. For instance, the evolutionary method may be hybridized with the local search algorithm. If the local search is based on exchanging one gene value to the other, the optimization of computation load brought by using PEACh effect may be significant as well. In this paper we also show that for some methods like standard GA the use PEACh is limited. However, its influence may also lead to statistically significant effectiveness increase.

Another interesting observation refers to computation load measurement. When an evolutionary method is applied to solve a practical problem, it seems reasonable to use PEACh effect if possible. However, when PEACh is used the FFE is not a fair computation load measure because the cost of computing a single fitness evaluation may be significantly different depending on the situation in which it is computed - if fitness is computed after a small genotype change (eg. after mutation) the cost will be low, in other cases it will be high. Thus, in such cases, FFE is not a reliable computation load measure.

VI. CONCLUSION

The objective of this paper was to check how significant may be the optimization of computation load expenses when PEACh effect is employed by methods applied to solve hard, practical problem. The presented results show that the influence of PEACh may be significant even if the method is not suitable to employ it (HEFAN 2.2). On the other hand, for methods using mechanisms like messy-coding, the efficiency increase may exceed 300%. The results supported by statistical tests point out that the result quality differences caused by PEACh are significant. Thus, for the considered methods FFE is not a fair computation load measure.

The key direction of future research is further investigation of possible PEACh utilization, for instance in hybrid methods using local optimization to improve their effectiveness. The use of PEACh may also enable significant effectiveness breakthrough for methods employing the Baldwin effect [16] as it may significantly reduce its computational costs. Finally, new techniques that use problem features for computation load reduction shall be identified and proposed.

ACKNOWLEDGMENT

This work was supported by the Polish National Science Centre (NCN) under Grant 2015/19/D/ST6/03115.

REFERENCES

[1] L. Fratta M. Gerla, L. Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," *Networks*, vol. 3, no. 2, 1973, pp. 97-133.

[2] D.E. Goldberg, B. Korb, K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, vol. 3, 1989, pp. 493-530.

[3] B. W. Goldman, W. F. Punch, "Parameter-less Population Pyramid," *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14)*, ACM, 2014, pp. 785-792.

[4] W. D. Grover, "Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking," Prentice Hall PTR, New Jersey, 2004.

[5] S.-H. Hsu, T.-L. Yu, "Optimization by Pairwise Linkage Detection, Incremental Linkage Set, and Restricted / Back Mixing: DSMGA-II," *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, ACM, 2015, pp. 519-526.

[6] D.R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, 2001, pp.345-383.

[7] M. M. Komarnicki, M. W. Przewozniczek, "The influence of fitness caching on modern evolutionary methods and fair computation load measurement," *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18)*, ACM, 2018, (in press).

[8] H. Kwasnicka, M. Przewozniczek, "Multi Population Pattern Searching Algorithm: a new evolutionary method based on the idea of messy Genetic Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 715-734, 2011.

[9] P.B. Myszkowski, M. Przewozniczek, M. Skowronski, "Constructive heuristics for technology-driven Resource Constrained Scheduling Problem," *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, 2015.

[10] P.B. Myszkowski, M. Skowronski, L. Olech, K. Oslizlo, "Hybrid Ant Colony Optimization in solving Multi-Skill Resource-Constrained Project Scheduling Problem," *Soft Computing*, vol. 19, issue 12, 2015, pp 3599-3619.

[11] M. Pioro, D. Medhi, "Routing, Flow, and Capacity Design in Communication and Computer Networks," Morgan Kaufmann Publishers, 2004.

[12] R. J. Povinelli, X. Feng, "Improving Genetic Algorithms Performance By Hashing Fitness Values," *Artificial Neural Networks in Engineering*, 1999, 399-404.

[13] M. Przewozniczek, K. Walkowiak, "Quasi-hierarchical Evolutionary Algorithm for Flow Optimization in Survivable MPLS Networks ," *Lecture Notes in Computer Science,* vol. 4707, Springer Verlag, 2007, pp. 330-342.

[14] M. Przewozniczek, R. Goscien, K. Walkowiak, M. Klinkowski, "Towards Solving Practical Problems of Large Solution Space Using a Novel Pattern Searching Hybrid Evolutionary Algorithm - An Elastic Optical Network Optimization Case Study" in Expert Systems with Applications," vol. 42, 2015, pp. 7781-7796.

[15] M. Przewozniczek, "Active Multi Population Pattern Searching Algorithm for Flow Optimization in Computer Networks - the novel coevolution schema combined with linkage learning," *Information Sciences*, vol. 355-356, 2016, pp. 15-36.

[16] M. W. Przewozniczek, K. Walkowiak, M. Aibin, "The evolutionary cost of Baldwin effect in the routing and spectrum allocation problem in elastic optical networks," *Applied Soft Computing*, vol. 52, 2017, pp. 843-862.

[17] M. W. Przewozniczek, "Problem Encoding Allowing Cheap Fitness Computation of Mutated Individuals," *Proceedings of 2017 Congress on Evolutionary Computation (CEC 2017)*, 2017, pp. 308-316.

[18] D. Thierens, P. A. N. Bosman, "Hierarchical Problem Solving with the Linkage Tree Genetic Algorithm," *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO'13)*, 2013, pp. 877-884.

[19] F. Zhang , X. Zheng, H. Zhang, Y. Guo, "A kind of topology aggregation algorithm in hierarchical wavelength-routed optical networks," *Photonic Network Communications*, vol. 9, 2005, pp. 167-180.