DOI: 10.15439/2015F411
ACSIS, Vol. 5

DiverGene: Experiments on Controlling Population Diversity in Genetic Algorithm with a Dispersion Operator

Anna Strzeżek', Ludwik Trammer', Marcin Sydow' "
'Polish-Japanese Institute of Information Technology
ul. Koszykowa 86, 02-008 Warszawa, Poland
"Institute of Computer Science, Polish Academy of Sciences, Warszawa, Poland
Email: astrzezek@gmail.com, ludwik@trammer.pl, msyd@poljap.edu.pl

***Abstract*—We present diverGene – a novel, diversity-aware population selection operator for genetic algorithm – to be used especially for particularly complex and multi-criteria optimisation problems. Genetic algorithm is one of the most known evolutionary algorithms for solving hard optimisation problems. Many attempts have been made to improve its convergence rate and quality of the result. In this paper we propose a novel extension of the selection operator that makes it possible to control the level of diversity in the population. We discuss its theoretical background, including its computational hardness and propose an efficient way of computing it. The approach is implemented and tested on three hard optimisation problems: Knapsack Problem, Travelling Salesman Problem and a relatively new Travelling Thief Problem that might be viewed as the composition of the latter two. We report experimental results that seem to indicate that the novel approach has a potential to improve the quality of the results for some hard optimisation problems.**

I. Introduction

In this paper we present `diverGene` – a novel, diversity-aware population selection operator for genetic algorithm. The idea is based on the concept of the *dispersion* of the solutions modelled by means of pair-wise dissimilarity between the solutions.

The proposed operator seems to be especially useful for particularly complex optimisation problems of multi-criteria nature, where the solution space should be intensively explored due to the potential variety and mutual non-similarity of potential good solutions to the problem.

A. Genetic Algorithm

The *genetic algorithm* [1] is a heuristic for finding satisfactory solutions to problems, using a process inspired by the natural selection. It does not guarantee to find the optimal solution and is intended to be used mainly in cases when finding the optimal solution directly would be too computationally expensive to be practical (more formally for NP-hard optimisation problems). In such cases search heuristics are often used to find a solution that is usually not the best possible one, but is still useful.

The process starts by randomly creating a set of feasible solutions to the problem. This set of solutions is called *population* and each solution in the population is called *an individual*, in reference to the biological origins of the algorithm. The initial state of the population is called *the initial (first) generation*. During the course of the algorithm execution, the subsequent generations are created, each one based on its immediate predecessor. While the solutions encoded in the initial generation are completely random, the quality of solutions in each subsequent generation should gradually get better, in an evolutionary fashion.

1) Selection: The process of selection is responsible for the decision on *which* individuals from the current generation should be used when creating the next generation (or, in other words, how many children should an individual have - if any). This step is essential to the process. Properly defined selection causes the best individuals to reproduce more intensively, resulting in a general increase of the solution quality in the population. On the other hand a selection that is *too* strong may lead to some problems - if the algorithm were to always simplistically select only a small subset of the very best solutions, there would be a high likelihood of losing the diversity and getting stuck in a local maximum. The problem is traditionally solved by introducing an element of randomness - the better the solution the greater the possibility it will be selected as a parent. As a consequence, other solutions also have a chance of being selected, while the overall solution quality generally increases.

2) Mutation Operator: The process of selection results in the increase of the *overall* solution quality in a population by generally eliminating weak solutions and reproducing stronger ones, but by itself it doesn't improve the quality of individual solutions. It's the mutation operator that enables the algorithm to explore other (possibly better) solutions. During the mutation stage there is a small chance of individual elements of a solution being randomly changed, resulting in a solution that is similar to one previously selected, but slightly different. Sometimes the changes are beneficial, but perhaps even more

This work is partially supported by Polish Ministry of Science and Higher Education grant N N519 578038.

often they make the solution weaker. During the subsequent selection stages the stronger solutions are more likely to be chosen than the weaker ones. As a consequence individuals with harmful mutations will eventually be eliminated, while individuals with beneficial mutations will become the parents for multiple newly created individuals, some of which will in turn be mutated, bringing the possibility of further improvements. The mutation operator corresponds to the concept of *random local search* in the solution space.

3) Cross-over Operator: Cross-over stage brings a possibility of two different solutions being combined. This allows for new individuals that include good parts of multiple previously known solutions. Without the cross-over stage there would be no sharing of good "ideas" within the population.

B. Motivation

As discussed in the "Selection" section above, too much focus on promoting only the best solutions in the current population during the selection stage (at the expense of alternative candidate solutions) may result in inadequate exploration of the space of possible solutions, or, equivalently in getting stuck in a local optimum.

On the other hand too little focus on choosing good result will result in a weak evolutionary pressure and population not getting better over time. The balance is traditionally achieved by giving better quality solutions a greater probability of being selected, but allowing other solutions to also be occasionally selected by random chance. While this method allows for some additional diversity it does not guarantee it.

In this paper, we investigate whether controlling the level of diversity within a population at the expense of a slight drop in the overall population quality would have a positive effect on the overall performance of the algorithm. In particualr we want to explore the effect of this aproach on problems containing multiple interdependent sub-problems, using the Travelling Thief Problem [8] as an example. It seems that increased population diversity would especially improve the performance of the genetic algorithm for such compound hard optimisation problems.

C. Contributions

The contributions of this paper are the following:

- we propose `diverGene` – a novel *diversity-aware* selection operator that makes it possible to control the balance between the exploration and exploatation of the solution space. The selection is based on both fitness and diversity among its members.
- we report development of a flexible software framework for testing the operator with various settings.
- we report experimental results on three hard optimisation problems: the Knapsack problem, the Travelling Salesman Problem and the Travelling Thief Problem (the last one being particularly noteworthy since it is composed of two interdependent sub-problems). The experiments concern the impact of the operator in various settings on the quality of the solutions.

D. Contents

We make a brief overview of the related works in Section II.

We give some theoretical background concerning the concept of diversity and, in particular, modeling it by means of *dispersion* in Section III. In this section we also present diverGene, our diversity-aware selection operator, and explain that exact computation of our operator is a hard computational optimisation problem itself by linking it to a known NP-hard optimisation problem.

Because of this, in Section IV, we explain how our selection operator can be efficiently computed with a known poly-time approximation algorithm.

Section V contains the specifications of the used benchmark optimisation problems and describes the some technical representation and implementation details.

In Section VI we report experimental results that are discussed in Section VII. We conclude in Section VIII.

II. Related Work

Genetic algorithm belongs to the class of evolutionary algorithms. First mentions of algorithm based on the mechanics of biological evolution dates back to the half of the previous century, but it became popular through the work of John Holland [1].

The concept of Diversity plays an important role in complex systems [2]. In particular, the diversity of population in biology is an important mechanism that supports better exploration of the environment. Therefore, the concept of diversity-awareness of population in biology is an inspiration for the domain of genetic algorithms to avoid a premature convergence (exploatation).

The Max-Sum Facility Dispersion Problem, that inspired our choice of a specific approach to diversify the population was studied in many papers in the domain of Operational Research. An example of such work is [3] that also discusses its computational hardness and efficient approximation algorithms for this problem. The connection of this problem to Web search result diversification with a new, parameterised objective function that takes into account a balance between the total value and pair-wise dispersion is discussed in [4] which also inspired our approach presented in this paper. Similar approach to diversification was proposed in the problem of semantic entity summarisation in [6] where another nature-inspired approach is applied to optimise the diversification problem itself. The study of the influence of the properties of the pair-wise dissimilarity function on the approximation factor in Facility Dispersion Problem is recently studied in [7].

The Travelling Thief Problem was introduced in [8] as a benchmark for testing heuristic algorithms on a problem that better resembles real life problems, that are often way more complex than well known “academic” benchmark problems like TSP or Knapsack. By its definition, the Travelling Thief Problem, is a kind of combination of the Knapsack Problem and TSP Problem.

III. diverGene: Diversity-aware Selection Operator

In this section we introduce diverGene – a modified selection operator that makes it possible to control the desired level of diversity in the population in genetic algorithm. In the classical genetic algorithm, the selection operator is based mainly on the fitness function of an individual. Our diversity-aware selection operator takes the fitness score into account, but additionally aims at guaranteeing some level of the diversity of solutions in the selected population.

More precisely, *diversity-aware selection* can be viewed as an optimisation problem itself and can be defined as to select a sub-population S of cardinality k out of the whole current population that maximizes the following parameterised objective function:

$$f_d(S) = (1-\alpha)\sum_{p\in S} f(p) + \alpha \sum_{p,q\in S} dissimilarity(p,q) \quad (1)$$

Where S is the selected, diverse sub-population and α is a parameter controlling the balance between the total $fitness$ (the $f()$ function) of the selected individuals and pair-wise $dissimilarity$ between them. $dissimilarity$ is a function returning a numerical value showing how different two individuals are. It's exact definition depends on the problem under consideration.

A. A Link to Max-Sum Facility Dispersion Problem

Now we are going to explain that maximising the objective function 1 used in the definition of our population diversity operator at the beginning of Section III is equivalent to some well-known optimisation problem that is NP-hard but for which there exists a fast approximation algorithm. Due to this we use this approximation algorithm for efficiently compute the population diversity operator.

More precisely, the diversity selection operator described in Equation 1 is inspired by the Facility Dispersion Problem [3] and its recent applications in Web Search Diversification Problem [4].

In Facility Dispersion Problem, the input consists of a complete, undirected graph $G(V,E)$, an edge-weight function $d: V^2 \rightarrow R^+ \cup \{0\}$ that represents *pairwise distance* between the vertices and a positive natural number $k \leq |V|$. The task is to select a k-element subset $S \subseteq V$ that maximises the objective function $dispersion(S)$ that represents the notion of *dispersion* of the elements of the selected set S.

Facility Dispersion Problem was studied in operational research for modeling the problem of selecting a set of mutually-distant (dispersed) locations for k obnoxious facilities like nuclear plants, ammunition dumps, etc.

In the most common variant of this problem, called Max-Sum Dispersion Problem, the $dispersion$ function to be maximised is defined as the total pair-wise distance between the selected locations (to be maximised):

$$dispersion(S) = \sum_{\{u,v\}\subseteq S} d(u,v) \quad (2)$$

The Max Sum Dispersion problem is NP-hard even if the distance function d is a metric, but in such case there exists a polynomial-time algorithm of approximation factor of 2 that was presented in [3].

Recently, the Max-Sum Facility Dispersion problem has new applications in the *Web Search Result Diversification Problem* where the selected items represent documents (or pieces of information) to be returned by the search system and d models the pairwise *dissimilarity* between the documents.

Given a set V of documents to be potentially relevant to a user query, a number $p \in N^+, k < |V|$, a *document relevance* function $w : V \rightarrow R^+$ and pairwise *document dissimilarity* function $d : V^2 \rightarrow R^+ \cup \{0\}$, the task is to select a subset $S \subseteq V$ that maximises the value of a properly defined *diversity-aware relevance function.* In [4] the following parameterised, bi-criteria objective function (to be maximised) is proposed as the diversity-aware relevance function:

$$f_{div-sum}(\lambda, S) = (k-1)\sum_{v\in S} w(v) + 2\lambda \sum_{\{u,v\}\subseteq S} d(u,v) \quad (3)$$

where $\lambda \in R^+ \cup \{0\}$ is a parameter that controls the diversity/relevance-balance.

In the same work it is observed that a proper modification of d to d' (Equation 4) makes the described problem of maximising $f_{div-sum}(\lambda, S)$ equivalent to maximising $\sum_{\{u,v\}\subseteq S} d'_\lambda(u,v)$, where:

$$d'_\lambda(u,v) = w(u) + w(v) + 2\lambda d(u,v) \quad (4)$$

Thus, it makes the *result diversification* problem described above equivalent to the Max Sum Dispersion problem for d'_λ.

Now, it is not hard to see that the problem of maximising our dispersion-aware fitness function f (for any value of α) defined in Equation 1 is equivalent to the problem of maximising the objective function in Equation 3 for appropriate selection of the value of λ, thus the problems are equivalent.

To sum up, as the consequence, the problem of maximising the diversity-aware population fitness function defined in Equation 1 is NP-hard, as being equivalent to Max-Sum Dispersion Problem, but any approximation algorithm used for Max-Sum Dispersion can be used to optimise our problem.

Due to this, in the experiments reported in this paper we use the 2-factor greedy approximation algorithm for Max-Sum Facility Dispersion problem described in [3] to efficiently solve our problem. This is described in Section IV.

IV. Dissimilarity selection

Our efficient implementation of diversity-aware selection operator is based on the algorithm mentioned at the end of the previous section. This approximation algorithm for finding subsets (in which weights of edges between those vertices are maximised) of vertices in a graph, can easily be adapted

for choosing a subset of individuals in a population that are most different from each other. One just needs to think of individuals as vertices and the level of dissimilarity between two individuals as the weight value on edges between the vertices representing them.
In our case we wanted to take into account both dissimilarity and fitness of the two individuals, as can be seen in the function defined in Equation 1. We utilised the link to the previously studied problems of Max-Sum Facility Dispersion and Web search diversification described in the previous section to define a $dissimilarity\_fitness$ function that combines dissimilarity between two individuals and their respective fitness values:

$$df(u,v) = fitness(u) + fitness(v) + 2\lambda dissimilarity(u,v)$$

We used the function to calculate values of edges between the individuals on the graph.
More precisely, the algorithm that we utilize to choose a k-element subset of individuals from the previous population to control its guaranteed level of diversity works in the following way:

1) Create a list with all possible pairs of individuals in a population.
2) Choose the pair with the highest $dissimilarity\_fitness$ score.
3) Remove all pairs containing one of the individuals from the chosen pair.

Repeat steps 2–3 $k/2$ times, where k is the desired size of the selected population. If k is odd, add an arbitrary final individual to the selected sub-population. This greedy algorithm guarantees the approximation factor of 2 to the solution of such defined optimisation problem [3].

A. Combining two types of selection

We create the new population of size N by combining the results of two kinds of selection - classical *roulette selection* and our own *dissimilarity selection* described in Section IV. First we select k individuals using dissimilarity selection and then achieve the desired population size N by selecting the remaining $N - k$ of the individuals using standard roulette selection. The result for small values of k can be viewed as classic selection enriched by a small pool of individuals retained for their uniqueness (in combination with their fitness, since $dissimilarity\_selection$ takes both into account).

V. Experimental Implementation

We used the Python programming language to create a flexible framework to test our novel diversity-aware operator on various optimisation problems.

In this Section we report experiments performed on two classic optimisation problems - Knapsack Problem and Travelling Salesman Problem. We also report additional series of experiments concerning the combination of the latter two – the Travelling Thief Problem. The framework is designed so that in the continuation work, the plugins with support for other optimisation problems can be easily added to it.

A. Knapsack Problem

The Knapsack Problem is one of the a classic NP-hard optimization problems. In this problem, there are n items. Each item has a value ($p_i \in Q^+$) and a weight ($w_i \in Q^+$). The capacity ($W \in Q+$) of the knapsack is limited.

The items should be picked so that their total value is maximized while their total weight does not exceed the knapsack capacity:

$$f(\bar{x}) = \sum_{i=1}^{n} p_i x_i, while \sum_{i=1}^{n} w_i x_i \le W$$

where $\bar{x} = (x_1, x_2, \ldots, x_n)$ and $x_i \in \{0, 1\}$ indicates whether the item is picked ($x_i = 1$) or not ($x_i = 0$).

1) Representing the Knapsack Problem in GA Framework: A solution is represented as a set of items in a knapsack. The most natural representation of any solution as a chromosome is in the form of a characteristic vector of the given subset of the items.

The mutation operator "flips" bits at random positions of the corresponding characteristic vector. Each position has an equal probability of being flipped (in our experiments the probability always equals $p_{km} = 0.01$). If the item was previously in the set it will be removed, otherwise it will be added to the set.

The cross-over operator exchanges the state between random items between two solutions. Each position has an equal probability of being exchanged (in our experiments the probability always equals $p_{kc} = 0.05$).

Both operators may result in a solution that exceeds the allowed capacity. In such case we fix the solution by removing random items from the set until the capacity is abided by.

The dissimilarity score is calculated by counting the number of items present in one solution but not present in the other (the *Hamming distance*).

2) Knapsack Dataset: The dataset for Knapsack problem was generated purposely for our experiments. Each instance contains 32 items with different values and weights (both ranging between 1 and 500). The knapsack capacity is set to 500.

B. Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is also a classic NP-hard optimization problem. In this paper, we consider a 2-dimensional euclidean variant. In this variant of TSP, there are n cities (c) with their coordinates (cx and cy). A salesman must visit each city exactly once and minimize the total length of the complete tour. The aim is to find a permutaion of the set containing all cities which minimizes the following equation:

$$f(\bar{c}) = \sum_{i=1}^{n-1} d(c_i, c_{i+1}) + d(c_n, c_1)$$

where $\bar{c} = (c_1, c_2, \ldots, c_n)$ represents the tour and $d(A, B)$ is a distance between the cities A and B:

$$d(A, B) = \sqrt{(Ax - Bx)^2 + (Ay - By)^2}$$

1) Representing TSP in GA Framework: The *ordered crossover* and *reverse sequence mutation* operators were used for TSP. The ordered crossover operator, presented by Goldberg in [5], is a two-point crossover. Given two random crossover points parent chromosomes are split into three parts. Child chromosome inherits the left and right parts from the first parent, and the middle part from the second one. The elemetns in the right and left parts are removed and rotated to avoid the duplication of elements [5].

In the reverse sequence mutation operator the sequence of elements between two randomly chosen positions is reversed.

The dissimilarity operator takes into consideration the number of different elements on the same positions in both chromosomes and compares it to the total number of elements. For each position, the values from both individuals are compared. If they differ, the counter is incremented. After analyzing the whole chromosomes the counter value is divided by the size of a chromosome (number of all cities in the instance).

2) TSP Dataset: The dataset for TSP experiments was taken from the TSPLIB library [9] containing sample instances for TSP. Berlin52 set was chosen [9] for the experiments. This set provides coordinates of 52 locations. The optimal solution to this problem is known, and it has the value of 7542.

C. Travelling Thief Problem

The *Travelling Thief Problem* is an optimisation problem related to the Knapsack Problem and the Travelling Salesman Problem (both described above). It was created to better model real-life problems, often more complex than the well-known benchmark NP-hard problems.

The Travelling Thief Problem consists of two interdependent sub-problems. The sub-problems can not be solved individually, because the solution to one sub-problem strongly influences the quality of the solution of the second sub-problem. To achieve satisfactory results one needs to solve both sub-problems at the same time, finding the best *combination* of solutions. This resembles the challenge often occurring in the case of solving complex real-life problems with multiple, often mutually contradictory, constraints.

We decided to include the Travelling Thief Problem in our experiments as it seems a natural next step of our work since it can be viewed as a specific *composition* of the two problems described above: the Knapsack Problem and TSP. Thus, the two components of the specification introduce a multi-criteria structure. Additional complexity of this problem is introduced by modeling the varying *speed*, that depends on the current contents of the knapsack, that influences the total cost of the solution, as explained below in this Section. Such a complex, multi-criteria optimisation specification seems to be a very natural domain of application for our diversity-aware population control approach as it enforces exploring multpile, mutually dissimilar alternatives for the solution.

There are several variants of the TTP problem. The one implemented by us, known as TTP_1, is most likely the most popular one. There are n cities (represented by a distance matrix) and m items (each having its own value, weight and set of cities in which it can be found). A thief has to visit each city exactly once, picking some items located in those cities and putting them in their knapsack (which has a maximum weight capacity W). Thief's speed depends on the weight of the knapsack:

$$V_C = V_{max} - W_C \frac{V_{max} - V_{min}}{W}$$

V_C represents the current speed, W_C represents the current weight of the knapsack and W represents the knapsack's maximum weight capacity. V_{max} and V_{min} are parameters describing the thief's maximum and minimum speed, respectively. The natural interpretation is as follows: the heavier the knapsack, the slower the thief. The speed is important, because the objective function that is to be maximized depends not only on the value of the picked items, but also on the total time of the travel, as the thief has to pay for the time when the knapsack is being used:

$$G(x, z) = g(z) - R \cdot t(x, z)$$

Where g is the sum of values of all items in the knapsack, R is the knapsack payment fee (rent) for the time unit and t is the function calculating the time of the travel for the given solution. This means the thief needs to maximise their profit - the value of the stolen items reduced by the knapsack fee.

A solution consists of a tour x and item picking plan z (recording the information concerning which items should be picked in which city - if it is picked at all). The G function strongly interconnects them, so they can not be optimised separately.

1) Representing TTP in GA Framework: TTP representation in GA framework is a union of previously described Knapsack and TSP representations. The dissimilarity value is calculated by counting the number of different picks - the items picked in the different cities at different times.

2) TTP Dataset: We used the dataset for TTP experiments taken from the webpage of the optimisation group of the University of Adelaide [10]. The data provided on this site was used as a benchmark set for competitions in 2014 and 2015. More precisely, the `Berlin52_n51_bounded-strongly-corr_01` dataset was chosen [10]. This set provides coordinates of 52 locations and 51 items, one in each city. The optimum solution to this instance is not provided by the creators of the dataset nor yet known, to the best knowledge of the authors.

VI. Experimental Results

In the experiments we measured the influence of the dissimilarity operator on the best solution of the objective function $f(S)$ for diffrent population sizes. The tests where performed in three groups for each problem:

1) The best solution of $f(S)$ as a function of k for 10, 25, 50 and 100 individuals in population with α set to 0.35 (Fig. 1, 2 & 3)

2) The best solution of $f(S)$ as a function of α for 10, 25, 50, 100 and 150 individuals in population with k set to 20% of population (Fig. 4, 5 & 6)
3) The best solution of $f(S)$ as a function of population size with α set to 0.35 and k set to 20% of population (Fig. 7, 8 & 9)

Each test was run 10 times and the median of all the 10 results was computed and reported.

A. Computational Platform

Since the computations involved in our experiments were quite extensive it was more natural to perform them on a specialised computational platform. Due to this, all calculations were performed on the online data science platform Sense [11]. Sense is a recently launched (released March 18, 2015) cloud platform for data science and big data analytics built by Sense, Inc. company. It provides an interface to multiple scripted analytic tools like Python, Node.js, SQL and many more, and allows to launch a new engine with dedicated CPU and RAM for every created job. It also makes sharing data, scripts and jobs and colaborating on the same project easy.

VII. Discussion of the Results

A. The Best Solution of $f(S)$ as a Function of k

For the Knapsack problem some improvements in quality of solutions can be observed only for small populations (Fig. 1). For 10 and 25 individuals using the diversity-aware selection operator increased the value of the best solution by 12-15%. For bigger populations (50 and 100 individuals), though, the positive changes are almost unnoticeable. A critical value of k can be also observed for all populations. Exceeding $k = 0.7N$, where N is the population size, causes a dramatic decrease of the best solution value. This could be interpreted as too much exploatation at the expense of exploration for this relatively easy[1] problem would deteriorate the performance.

More significant positive impact of diversity operator on the quality of solution can be observed for TSP (Fig. 2). For all population sizes the results were improved by 30-35% when 15-35% of individuals were selected using the diversity-aware selection operator. The best solution decrease starts earlier than in the Knapsack problem but is not so rapid as in the case of Knapsack - in the worst case the percentage decrease between the best value of the objective function without and with diversity-aware selection operator is about 8% while in the Knapsack problem it is between 30-40%.

In the TTP experiments (Fig. 3) even stronger positive impact of our diversity operator on the solution quality can be observed. It reaches up to 40-45%. Importantly and interestingly, the peak representing the best solutions is shift towards the case when more individuals are being selected using our diversity-aware selection operator (about 70-75%). After reaching $k = 1N$ the collected results are the same or slightly worse then for computations without diversity-aware selection operator. That seems to indicate that for this, particularly hard and complex, optimisation problem the population diversity can only improve the performance of the algorithm while it almost *never* makes the quality worse.

[1]Remind that Knapsack is a relatively “easy” computational problem among the NP-hard problems as there exists a pseudo-polynomial dynamic programming algorithm for it, what excludes it from the class of, so called, strongly NP-hard problems

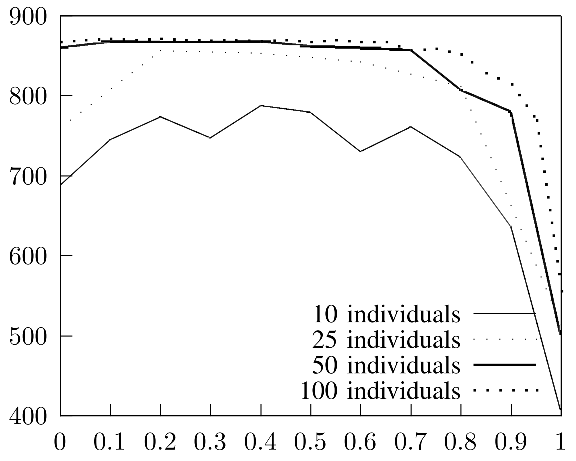

Fig. 1. $f(S)$ as a function of k for Knapsack problem.

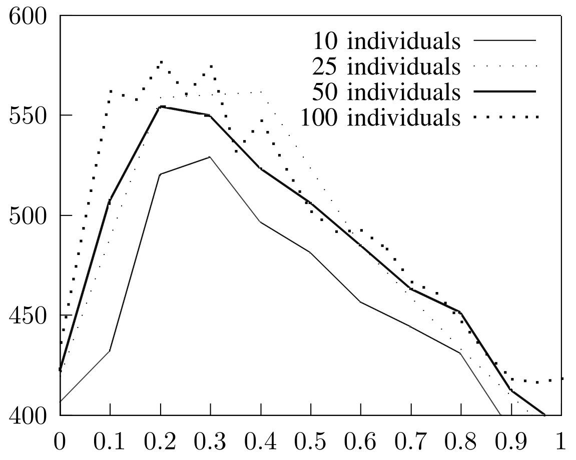

Fig. 2. $f(S)$ as a function of k for TSP.

B. The Best Solution of $f(S)$ as a Function of α

Experiments performed for the Knapsack problem (Fig. 4) show that changes of α parameter have a positive impact on the best solution value only for the smallest tested population composed of 10 individuals (28% increase for $\alpha = 0.15$). For bigger populations adding $dissimilarity$ to $f(S)$ has no or only negative impact on the results.
For our current experimental settings, the TSP (Fig. 5) and TTP (Fig. 6) tests show no clear dependence between the results and changes of α. This is an interesting signal that seems to be counter-intuitive (compared to the results concerning the dependence on the value of k presented above) and should be

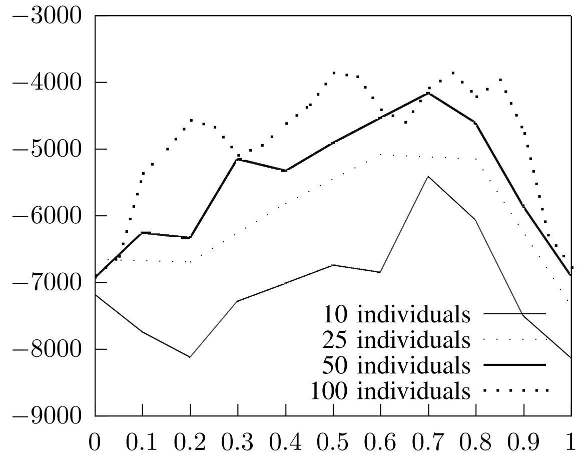

Fig. 3. $f(S)$ as a function of k for TTP.

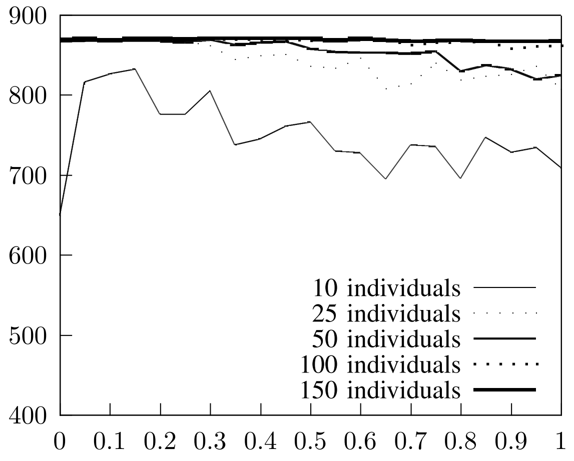

Fig. 4. $f(S)$ as a function of α for Knapsack problem.

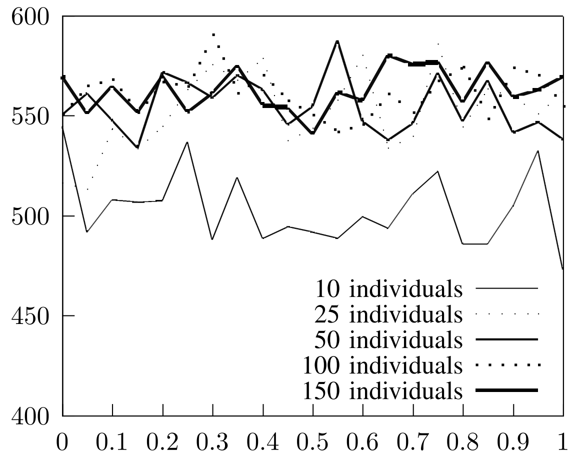

Fig. 5. $f(S)$ as a function of α for TSP.

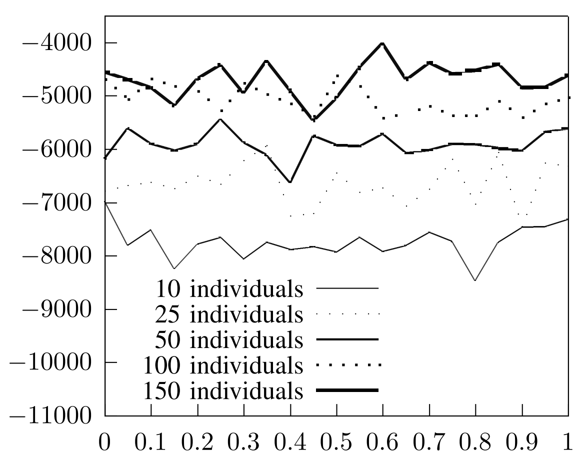

Fig. 6. $f(S)$ as a function of α for TTP.

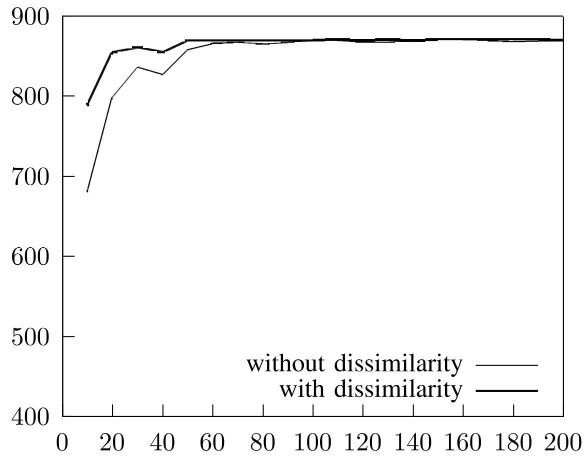

Fig. 7. $f(S)$ as a function of $|S|$ for Knapsack problem.

further examined in the continuation work.

C. The Best Solution of $f(S)$ as a Function of Population Size

Analysis of the last group of tests confirms the results observed in the first series of experiments (concerning the dependence of the solution quality on k). Namely, improvements in Knapsack problem solution (Fig. 7) are visible only for small populations and there is no change for bigger populations. For TSP (Fig. 8) a positive impact can be observed for all tested population sizes (10-200 individuals). The best solutions are equally improved throughout all the tested populations.

TTP results (Fig. 8) show, that for smaller populations there is a small postitve or even in some cases a negative impact on the objective function value. For bigger populations a positive impact is visible. It increases slightly with population size.

Most importantly, the reported experimental results clearly indicate that introducing diversity to the solution population can noticeably increase the quality of the solutions.

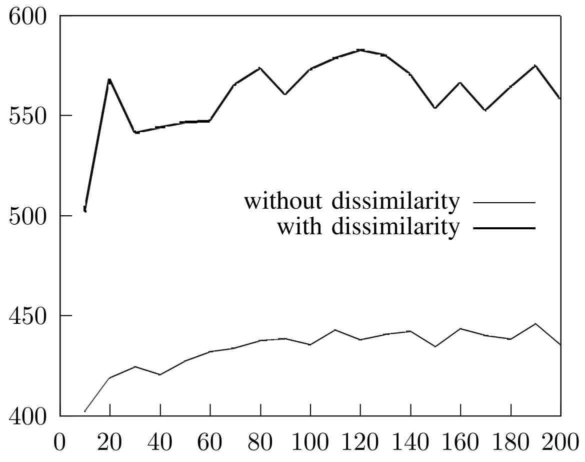

Fig. 8. $f(S)$ as a function of $|S|$ for TSP.

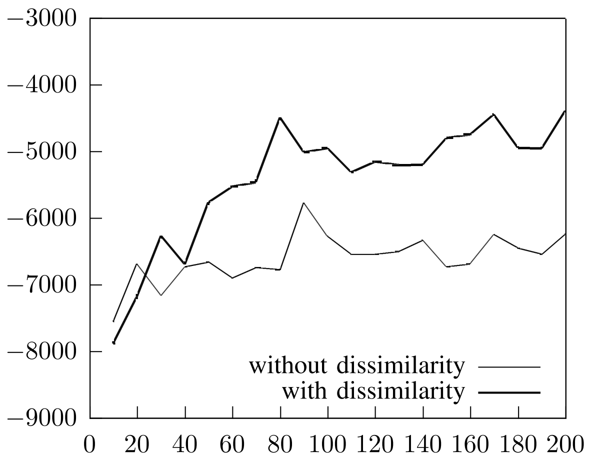

Fig. 9. $f(S)$ as a function of $|S|$ for TTP.

VIII. Conclusions and Future Work

The experiments reported in this paper clearly indicate that controlling the population diversity in genetic algorithm in the way we proposed in this paper by using `diverGene` is a promising technique. In particular, picking a part of population using the diversity-aware selection operator can significantly improve results of the objective function $f(S)$ for the variant of the TSP problem that we studied and, even more significantly, for the TTP problem. There is also an interesting observation, that seems to be natural and intuitive, that the more complex and harder the optimisation problem the more improvement can be achieved by applying our diversity-aware selection operator.

More tests should be performed for the studied problems, including examining other implementations of dissimilarity operator and other settings.

Another interesting direction could be to study the convergence rate as the function of the population diversity.

It is a bit surprising that we did not observe much dependence of the quality of the solution on the value of the α parameter. This should be further investigated in the future work.

It would be also interesting to introduce the self-adaptation mechanism to our approach. More precisely, to make the algorithm itself dynamically controlling the values of the diversity-aware operator parameters (e.g. k and α) to better control the exploration/exploatation balance.

IX. Acknowledgements

This work is partially supported by Polish Ministry of Science and Higher Education grant N N519 578038.

References

[1] John H. Holland *Adaptation in Natural and Artificial Systems.* University of Michigan Press,1975

[2] Scott E. Page. Diversity and complexity. *Primers in Complex Systems.* Princeton, NJ: Princeton UniversityPress. x, 291 p. $ 19.95, 2011

[3] R. Hassin, S.Rubinstein, A. Tamir, “Approximation algorithms for maximum dispersion”, *Operations research letters* vol. 21/3, 1997, pp. 133–137.

[4] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*, WWW ’09, pages 381–390, New York, NY, USA, 2009. ACM.

[5] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[6] W.Kosiński, T.Kuśmierczyk, P.Rembelski, M.Sydow "Application of Ant-Colony Optimisation to Compute Diversified Entity Summarisation on Semantic Knowledge Graphs", Proc. of International IEEE AAIA 2013/FedCSIS Conference, Annals of Computer Science and Information Systems, Volume 1, pp. 69-76, ISSN 2300-5963, ISBN 978-1-4673-4471-5 (Web), 2013

[7] M.Sydow "Approximation Guarantees for Max Sum and Max Min Facility Dispersion with Parameterised Triangle Inequality and Applications in Result Diversification" (extended journal version) Mathematica Applicanda Vol. 42, no. 2, pp. 241-257, DOI: 10.14708/ma.v42i0.547, Print ISSN: 1730-2668; On-line ISSN: 2299-4009, Polish Mathematical Society, 2014

[8] M. R. Bonyadi, Z. Michalewicz, L. Barone (2013, June). The travelling thief problem: the first step in the transition from theoretical problems to realistic problems. In Evolutionary Computation (CEC), 2013 IEEE Congress on (pp. 1037-1044). IEEE.

[9] http://www.iwr.uni-heidelberg.de/groups/comopt/software/ /TSPLIB95/tsp/berlin52.tsp.gz

[10] http://cs.adelaide.edu.au/ optlog/CEC2014COMP_InstancesNew/berlin52-ttp.rar

[11] http://wwww.sense.io

[12] Mohammad Reza Bonyadi, Zbigniew Michalewicz and Luigi Barone The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. *IEEE Congress on Evolutionary Computation.* IEEE, pp. 1037–1044, 2013