T-EA: A Traceable Evolutionary Algorithm

Cristian Ramirez-Atencia
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany
Email: cristian.ramirez@ovgu.de

Tobias Benecke
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany
Email: tobias.benecke@st.ovgu.de

Sanaz Mostaghim
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany
Email: sanaz.mostaghim@ovgu.de

***Abstract*—In this paper, the influence of the initial population into successive generations in Evolutionary Algorithms (EAs) is studied as a problem-independent approach. For this purpose, the Traceable Evolutionary Algorithm (T-EA) is proposed. This algorithm keeps track of the influence of the individuals from the initial population over the generations of the algorithm. The algorithm has been implemented for both bit-string and integer vector representations. In addition, in order to study the general influence of each individual, new impact factor metrics have been proposed. In this way, we aim to provide tools to measure the influence of initial individuals on the final solutions. As a proof of concept, three classical optimization problems (One Max, 0/1 Knapsack and Unbounded Knapsack problems) are used. We provide a framework that allows to explain why some individuals in the initial population work better than others in relation with the corresponding fitness values.**

***Index Terms*—Traceability, Evolutionary Algorithm, Initial Population**

I. Introduction

Evolutionary Algorithms (EA) can be used to deal with a large variety of complex NP-hard problems optimization or problems with no analytical forms such as black-box optimization problems or simulation-based problems [1].

Inspired by biological evolution, EAs use search mechanisms based on mutation, recombination and selection operators. These mechanisms are applied to individuals of a population (candidate solutions) for a number of generations until a stopping criterion is met. One of the main factors when designing an EA is the seeding of the initial population, which can largely influence the performance of the algorithm [2]. Population initialization is an important task which can affect convergence speed and therefore the quality of final solutions.

In the last decade, many problem-based methods have been developed to seed the initial population [3], [4]. In addition, general methods for initialising the population without a priori knowledge have been designed to accelerate the algorithms. Most of these approaches are based on quasi-random generators [5], opposition-based learning [6], gap filling [7] or combining gap filling with extending the edges of the front [8], among the others. Although these methodologies provide a great improvement to the algorithmic design, there is little theoretical work in terms of the convergence behaviour [9], [10]. In this paper, we aim to provide a new methodology to better understand the details of the search process such as the contribution of the individual properties and the components of EAs to the search. Our major focus is on tracing the influence of both initial solutions and operators on the final solutions.

In order to achieve this objective, a new algorithm called Traceable Evolutionary Algorithm (T-EA) is proposed. This algorithm records traces of the individuals in the initial population over the evolutionary process. In this way, it is possible to measure the amount of the influence of the predecessors of an individual in the final (or intermediate) generation. Tracking genes through historical markings was already proposed in NEAT [11], but in the context of performing artificial synapsis taking into account homologies, while in this paper the genes are traced for analytical purpose.

In this paper, we consider both bit-string and integer vector representations. In addition to T-EA, we propose new metrics to measure the influence of initial individuals on the final solutions. In order to provide a proof of concept with the proposed method, we examine our algorithm on three classical optimization problems (One Max, 0/1 Knapsack and Unbounded Knapsack problems). For each of these problems, different initial populations are used to check the behaviour of T-EA. The various initial populations contain different percentages of the initial individuals in terms of quality. Using the proposed metrics, the obtained results are analysed.

This paper is structured as follows. Section II presents T-EA and the corresponding operators. In Section III, the different impact factor metrics are proposed to measure the influence of initial individuals. Section IV shows the experimental setup and evaluation of the influence of initial individuals into final solutions for the selected problems and initial populations. Finally, in Section V, some conclusions of the work and future lines of research are provided.

II. Traceable Evolutionary Algorithm

For tracing the influence of the initial population to the final generation (population in the last generation), we propose the Traceable Evolutionary Algorithm (T-EA). In this algorithm, individuals store the information regarding their predecessors.

A. Encoding

More precisely, we add an additional vector with the same size as of the chromosome vector to each individual. This vector contains the IDs of the individuals (called trace ID) from the initial population. In this case, in the initial population, each individual puts its corresponding trace ID into

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

all the elements of the trace vector (e.g. for *Individual 1*, the elements of the trace vector are equally initialized to 1; for *Individual 2*, they are initialized to 2, and so on).

The goal is to propagate the trace IDs over the generations. In this case, every time a recombination and/or mutation are performed, the corresponding trace IDs of the genes have to get updated. An example of a traceable individual with integer representation is presented in Figure 1. In this paper, we will focus on two bit-string and integer vector representation and adapt the crossover operator for trace IDs in T-EA.

Genome	3	5	1	2	2	1
Trace	2	4	4	3	2	2

Fig. 1: Example of an individual in T-EA consisting of a genome and a trace IDs.

B. Crossover

Once a crossover is applied to a chromosome of an individual and the corresponding trace IDs are exchanged accordingly. This is referred as *traceable crossover*. In this way, the trace IDs from the initial population are preserved in the successive generations which can easily be used to observe the influence of the predecessors on an individual. Figure 2 shows an example of a 2-point crossover operator. It can be observed that the trace IDs keep track of the original genes.

Parent 1	3	4	0	0	2	1
	2	3	1	1	2	2
Parent 2	2	5	1	2	4	0
	4	4	4	3	3	5
Child 1	3	5	1	2	2	1
	2	4	4	3	2	2
Child 2	2	4	0	0	4	0
	4	3	1	1	3	5

Fig. 2: Example of a 2-point crossover in T-EA. Similar to the genes in the chromosome, every trace ID between the two crossover points is swapped between the parent individuals.

C. Mutation

Considering a mutation operator in the EAs such as uniform mutation, the genes are usually altered randomly and therefore the original information from the older generations get lost. This is valid for both binary and integer encodings. In T-EA, we propose a *traceable uniform mutation* such that every time a gene is mutated, we change the corresponding trace ID to a new mutation identifier. We use a global counter to generate the new trace IDs. With this, the mutations performed over the individuals during the algorithm are also traced. Therefore, it will be easily observable which mutations were most relevant for the final solutions. Figure 3 shows an example of this traceable uniform mutation, where genes 2 and 5 are selected to be mutated. As presented in the example, new trace IDs are created for these genes (m_1 and m_2). The identifiers in these mutation traces show that these have been the first and second ever mutated genes in the algorithm. In later mutations, the trace IDs would be m_3, m_4, and so on.

Parent	3	5	1	2	2	1
	2	4	4	3	2	2
Child	3	3	1	2	6	1
	2	m_1	4	3	m_2	2

Fig. 3: Example of uniform mutation in T-EA. For every randomly mutated gene, its trace ID is converted into a new mutation ID (m_1, m_2). A global counter generates new IDs.

With these traceable operations, T-EA is able to keep track of the individuals in the initial population. Additionally, we are able to follow the newly generated genes with the mutation operator to the final generation of the algorithm.

III. Impact Factor Metrics for T-EA

In this section, we provide new metrics to measure the influence of the initial individuals on the final population. The final population of T-EA contains the vectors of the trace IDs for each individual including the information regarding the influence of the initial population and the mutation traces. In order to describe the impact from an individual of the initial population on the final solutions, we propose the following three Impact Factor (IF) metrics: counting-based IF, fitness-based IF and entropy-based IF.

For formally defining the three IFs, the following notation is used: n denotes the number of individuals in a population (we suppose that populations of different generations have the same size). m denotes the size of the chromosome, i.e. the number of genes per individual. As this work deals with bit-string and integer vector representations, it is supposed that individuals have fixed sizes. P is the number of generations which additionally represents the index of the last generation.

The notations for individuals, populations and generations is as follows: the set G_i defines the population in the i-th generation of the algorithm. For each individual H of a specific population, the matrix $X^H = [x_j^H | j \in 1..m]$ is defined. Each element of the matrix represents the genes of that individual, so $G_i = ([X^H | H \in 1..n])_i$. Therefore, $(x_j^H)_i$ represents gene j of individual H in the population of generation i.

The fitness function $f : \mathbb{Z}^m \rightarrow \mathbb{R}$ is defined for bit-string and integer vector representations of individuals. In addition, a function $t : Gene \rightarrow Trace$ is defined to obtain the trace ID of the initial individuals and the mutation that generated the gene. In order to check whether the trace IDs of a specific gene correspond to an individual k from the initial population or not, the following function is defined:

$$T_k((x_j^H)_i) = \begin{cases} 1 & \text{if } t\left((x_j^H)_i\right) == k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A. Counting-based IF metric

The counting-based IF IF_C computes the influence of each individual k from the initial population on the final population by summing the occurrences of the trace IDs of k in the final population. This is then normalized by dividing the sum by the population size n and the chromosome size m:

$$IF_C(k) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} T_k\left((x_j^i)_P\right) \tag{2}$$

It can be noted that the influence of the mutation can additionally be computed using this metric by counting all the mutation trace IDs (or just a specific mutation trace ID if the influence of just one of these mutations is subject to study).

IF_C directly represents the influence of the first generation on the last in T-EA. However, it does not consider the fitness (quality) of the individuals.

B. Fitness-based IF metric

In order to additionally consider the impact of the initial solutions and the mutation on the quality of the solutions in the final population, we propose the Fitness-based IF metric IF_F. This metric takes the fitness value for each individual in the final population into account. IF_F is calculated in a similar way as IF_C and additionally multiplies the trace IDs by the corresponding fitness values. In this way, the good individuals from the initial population have a larger impact on the final population.$IF_F$is normalized by dividing it with the sum of all the fitness values in the final population:

$$IF_F(k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} T_k\left((x_j^i)_P\right) \cdot f((X^i)_P)}{n \cdot m \cdot \sum_{i=1}^{n} f((X^i)_P)} \tag{3}$$

Similar to counting-based IF, the fitness-based IF can also be used to measure the influence of the mutation operations by counting the number of mutation trace IDs.

C. Entropy-based IF metric

The counting-based IF favors the trace IDs which are present multiple times in the same individual. Other initial individuals that survived, but only in a few genes, are then underrepresented. The Entropy-based IF metric is designed to measure the influence of such initial individuals, as they have been able to survive until the last generation.

This metric takes into account the probability of a trace ID k appearing in a specific gene j

$$P_{k,j} = \frac{1}{n} \sum_{i=1}^{n} T_k((x_j^i)_P) \tag{4}$$

Based on the Shannon-Entropy [12], we define:

$$H\left(\Psi_j^g\right) = -\sum_{k \in \Psi_j^g} P_{k,j} \cdot log_2(P_{k,j}) \tag{5}$$

Where H is the entropy for gene j in generation g, and Ψ_j^g represents the trace IDs (including mutation trace IDs) that are present in gene j for all the individuals of the population in generation g. This entropy defines a value in $[0, 1]$ representing the importance of the gene, or how much information this gene provides to the solution. The entropy-based IF metric IF_E is presented in equation 6, where the previous IF_F metric has been extended by multiplying the entropy factor. As this entropy is in the range $[0, 1]$ and can be zero, we add 1 to avoid division by 0. As before, the metric is normalized by dividing with the sum of the entropy for every gene.

$$IF_E(k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} T_k\left((x_j^i)_P\right) \cdot f((X^i)_P) \cdot (1 + H\left(\Psi_j^P\right))}{n \cdot m \cdot \sum_{i=1}^{n} f((X^i)_P) \cdot \sum_{j=1}^{m} (1 + H\left(\Psi_j^P\right))} \tag{6}$$

IV. Proof of concept

The proposed T-EA and the IF metrics are used in the following experiments for a proof of concept. We intend to study their usability on different scenarios.

Three classical optimization problems are used for this purpose. We study two optimization problems based on bit-string representations (One Max and 0/1 Knapsack) and one based on integer encoding (unbounded knapsack). The goal of the One Max problem [13] is to maximize the number of 1s in a bit-string with a given length. The 0/1 knapsack problem [14] consists of several objects with specific weight and profit. The goal is to insert as many as possible objects into the knapsack with a fixed capacity. A possible solution of such a problem is a bit-string representing if an object is inserted (1) or not (0). The length of the bit-string corresponds to the number of objects. In the unbounded knapsack problem [15], there are unlimited elements of each object, and the knapsack is meant to be filled without over-passing the capacity. The solution of such a problem is represented by an integer vector with the length equal to the number of objects.

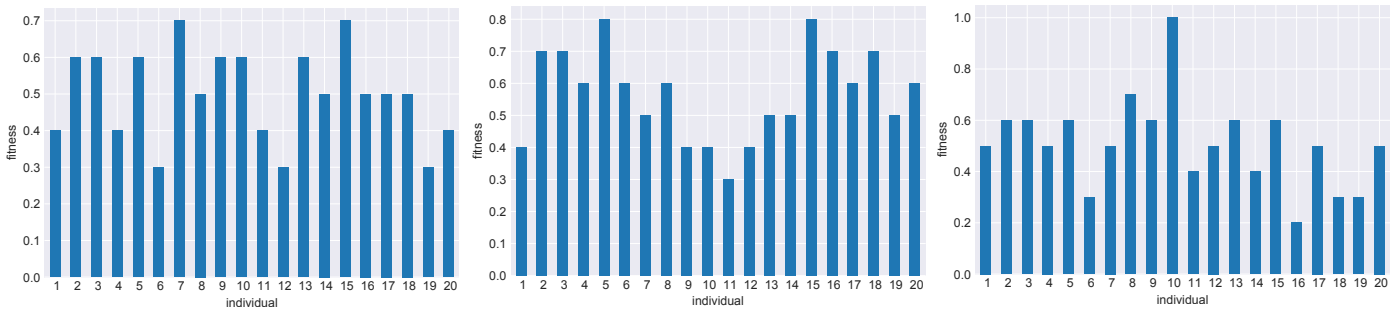
For testing T-EA, we take three different initial populations which are randomly generated. These populations have various properties: a large portion with high fitness value (Population 1), uniformly distributed fitness values (Population 2), and one dominating individual in terms of fitness value (Population 3).

A. Experimental setup

The setup used in T-EA for the experiments consists of a population size of 20 individuals, the traceable 2-point crossover and traceable uniform mutation are used. The mutation probability is set to 1%. A tournament selection of size 4 is used, and the algorithm is run for 20 generations. Each of the tests is repeated 31 times with different seeds, and the median and interquartile range are computed.

For the One Max problem, a bit-string of length 10 is used. Three different initial populations with various percentages of fitness values have been generated as presented in Figure 4.

The 0/1 knapsack problem in these experiments consists of 10 objects to be inserted in a knapsack of capacity 269. The weights and profits of these objects are presented in Table I. To deal with the capacity constraint of the knapsack, the parameter free constraint handling method [16] has been used. In this method, the worst fitness is defined as 0 and so invalid



(a) Population 1. (b) Population 2. (c) Population 3.

Fig. 4: Fitness of the individuals in the three proposed initial populations for the One Max problem.

solutions have negative fitness (the weight constraint value). Again, three different initial populations with corresponding fitness values as shown in Figure 5 are taken.

TABLE I: Weights and profits of the objects in the 0/1 knapsack problem and in the unbounded knapsack problem.

0/1 Knapsack		
Object	Profit	Weight
1	55	95
2	10	4
3	47	60
4	5	32
5	4	23
6	50	72
7	8	80
8	61	62
9	85	65
10	87	46

Unbounded Knapsack		
Object	Profit	Weight
1	33	15
2	24	20
3	36	17
4	37	8
5	12	31

In the unbounded knapsack we define 5 types of objects to be packed in a knapsack of capacity 80. The weights and profits of these objects are presented in Table I. The parameter free constraint handling is also used here. Figure 6 shows the three initial populations. In this test problem, the number of feasible solutions in the initial population is much smaller than in the 0/1 knapsack problem.

B. Experimental results

The results obtained after running T-EA with the different problems proposed are studied one by one as follows:

1) One Max problem: Figure 7 illustrates the results for the One Max problem with the three initial populations. For each pair of initial population and IF metric, a box plot is provided to represent the IF metric value for each individual of the initial population and the mutation operations.

For the Population 1, Figure 7a shows that, according to IF_C, *Individual 7* was the most influential, with a median of $IF_C(7) = 30\%$, followed by *Individual 15*. This makes sense as these individuals have the highest fitness values in the initial population. Mutation in general got a median $IF_C(Mut) = 10\%$, becoming the third most influential component. *Individuals 5* and *10* had some influence in some of the runs, but their median still remains near 0. *Individuals 2, 3, 8, 9, 13, 14, 17* and *18* appeared very few times, as they are outliers in the box plot, meaning that they have less than 8 appearances in the runs. In some of these cases the influence was large, like for example *Individual 2* who had $IF_C(2) = 50\%$ in one run. The rest of the individuals did not influence the final solutions, as they have low fitness values, so they probably were lost in most tournament selections.

Looking at the IF_F (Figure 7b) for this initial population, the results are quite similar. That means that either fitness values of the final solutions are similar, or the influence is well distributed so all the surviving individual traces appear in most solutions. In the case of IF_E (Figure 7c), the results are also quite similar except for some outliers whose influence may be higher due to the entropy as they appear in more "difficult" genes (a gene that is difficult to get its best value).

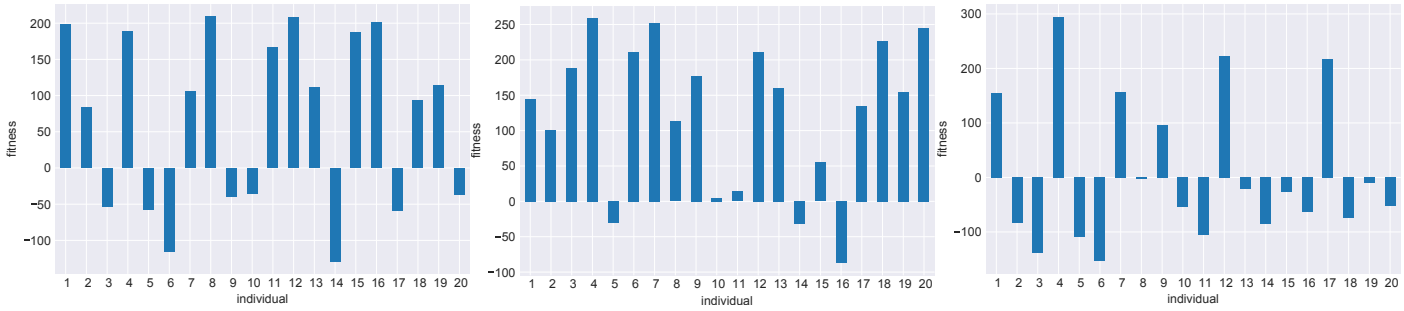
The results obtained for Population 2 (Figures 7d-7f) show a similar behaviour, as the two individuals with highest fitness had the highest influence. Nevertheless, the influence of *Individual 5* was higher than the influence of *Individual 15* (more than double) in median. It is observed that there are many outliers with large influence. It may be that in different executions, different individuals were used as a backup for the "difficult" genes in *Individuals 5* and *15*.

In Population 3 (Figures 7g-7i), the optimal solution was already in this initial population (*Individual 10*), so it is clear that this individual has the highest influence, having almost $IF_C(10) = 100\%$. *Individual 8*, who had the second best fitness also got some influence into the final solutions.

As the One Max problem is an easy and straightforward problem, it is common that most behaviours in the influence of the final solutions are expected, but still sometimes there could be unexpected individuals that influence the final solutions due to the stochastic nature of EAs.

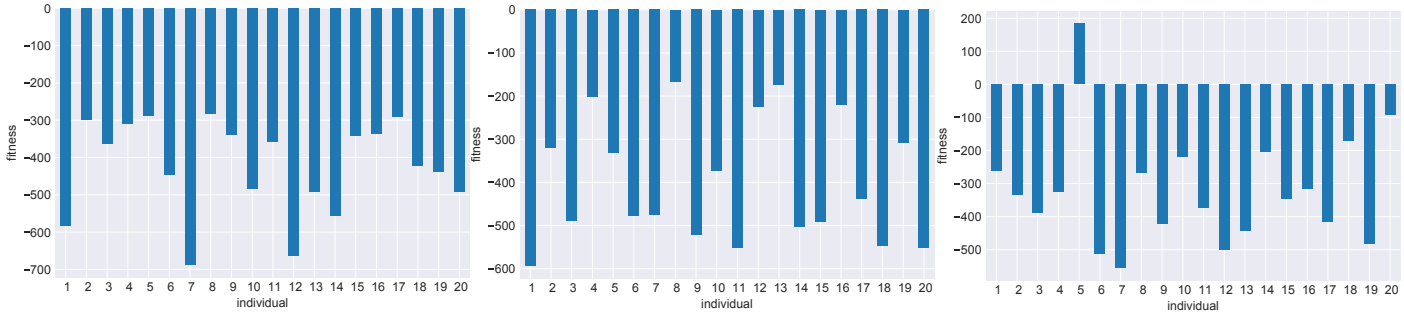
2) 0/1 knapsack problem: Figure 8 shows the results for the 0/1 knapsack problem with the three initial populations proposed and the three IF metrics as box plots.

In the results obtained for Population 1 (Figures 9a-9c), it is interesting that, although *Individuals 8* and *12* have very similar fitness, the first one had a huge influence on the final solutions, around $IF_C(8) = 75\%$, while *Individual 12* influence is near 0, except for some outliers. This indicates



(a) Population 1. (b) Population 2. (c) Population 3.

Fig. 5: Fitness of the individuals in the three proposed initial populations for the 0/1 knapsack problem.



(a) Population 1. (b) Population 2. (c) Population 3.

Fig. 6: Fitness of the individuals in the three proposed initial populations for the unbounded knapsack problem.

that these two individuals do not combine well, or even that *Individual 12* is a deceptive individual. *Individual 15*, on the other hand, although it has a worse fitness than others, such as *Individuals 1* or *16*, obtained a bigger influence in many cases, probably as a complementary of *Individual 8*. Although IF_F and IF_E got very similar results to IF_C, it is appreciable how the influence of some outliers is increased or decreased, even generating new outliers, e.g. in *Individuals 12* or *16*.

For Population 2 (Figures 9d-9f), the individuals with the best fitness (*Individuals 4, 7* and *20*) got the highest influence, with *Individual 4* being highly influential in most cases. Although *Individual 20* was not very influential, when considering the IF_E metric, its interquartile range became higher. That may represent that this individual may have good genes that are not common with *Individuals 4* and *7*. It is interesting to observe how other individuals which have fitness values not as good as the previous ones, but still quite good, were almost never considered, just in some outlier runs.

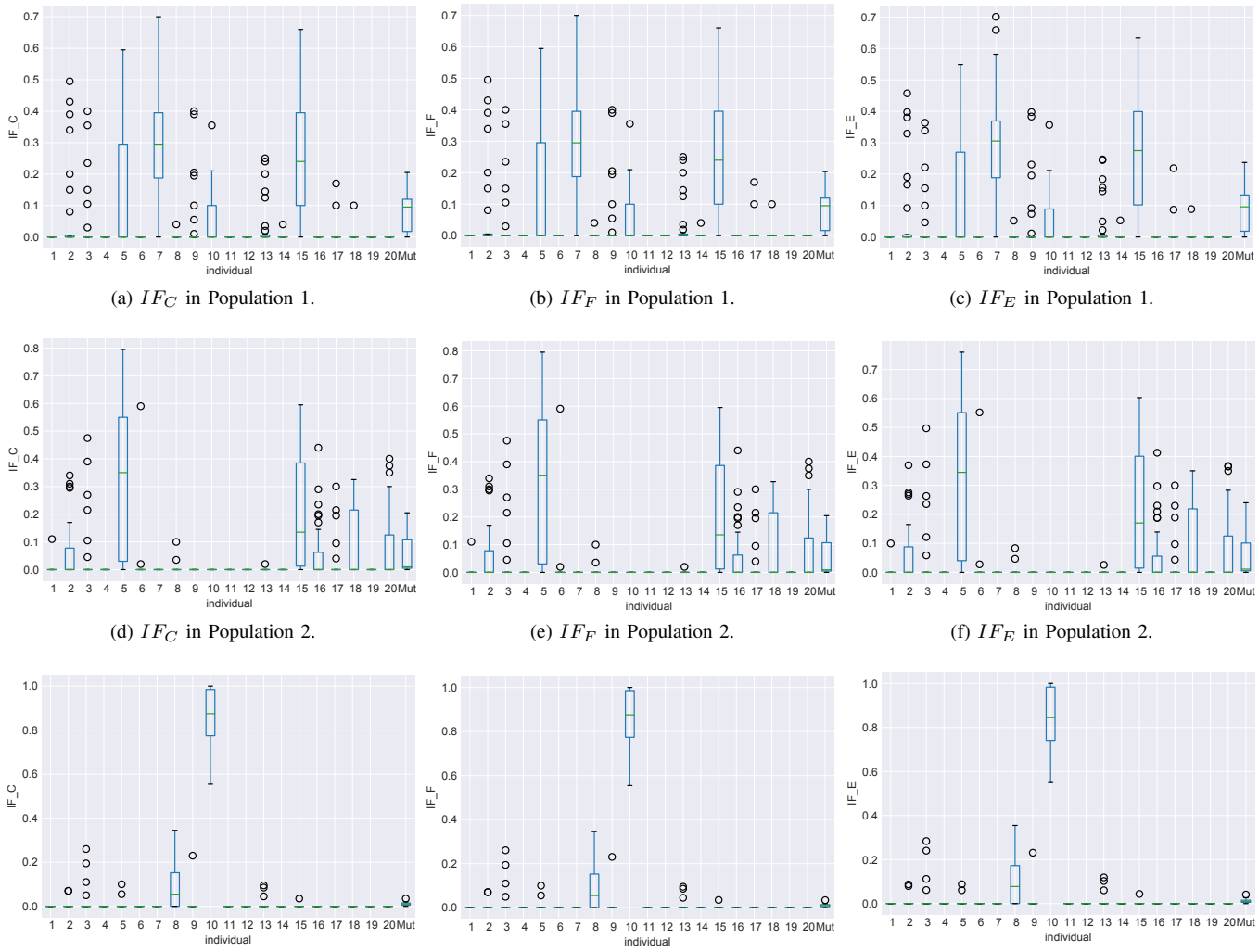
In the case of Population 3 (Figures 9g-9i), solution 4 seems to have reached the best fitness value and it is in most cases the highest influence for all the solutions, being near $IF_C(4) = 100\%$. The rest of individuals have a very low and outlier influence, even the mutation was useless in this case.

In the cases studied for the 0/1 knapsack problem, we have observed that a dominating solution in terms of the fitness value, tends to become the most influential individual and overtake the rest of individuals.

3) Unbounded knapsack problem: The unbounded knapsack problem was the integer representation tested in this work. The results are presented in Figure 9. As this problem is much more complicated than the previous ones, the results differ from the previous cases.

In the results obtained for Population 1 (Figures 9a-9c), there is not just one individual which became the most influential, but instead several individuals (*5, 8, 17* especially) had a moderate influence. This is due to the fact that all of the individuals in the initial population were infeasible solutions. In terms of the IF_F, it can be observed that the influence of *Individual 8* is larger than its impact with IF_C, as this individual may have appeared in solutions with better fitness than others. Nevertheless, for the IF_E metric, the behaviour is exchanged and *Individual 8* becomes less influential while *Individuals 5* and *17* gain some influence. This behaviour is due to those two individuals appearing in diverse genes while *Individual 8* always take over the same genes, becoming the only influence for those genes; and so for IF_E it is more relevant those diverse genes than the static ones. It is also very interesting to see that the mutation has a considerable influence according to IF_C but becomes irrelevant for IF_F and IF_E. The main reason seems to be that these mutations did not generate good solutions in terms of fitness values.

Population 2 (Figures 9d-9f) presents a similar behaviour,



(a) IF_C in Population 1. (b) IF_F in Population 1. (c) IF_E in Population 1.

(d) IF_C in Population 2. (e) IF_F in Population 2. (f) IF_E in Population 2.

(g) IF_C in Population 3. (h) IF_F in Population 3. (i) IF_E in Population 3.

Fig. 7: Results obtained for the IF_C, IF_F and IF_E metrics for each initial individual and the mutation operation in the One Max problem with the different initial populations.

as there is no feasible solution among the individuals in the beginning. *Individuals 8* and *13*, the ones with higher fitness, became the most influential ones, especially *Individual 8*. It can also be observed that its influence is highlighted by IF_F, which may indicate that it obtained better results in terms of fitness for the solutions that were generated from it.

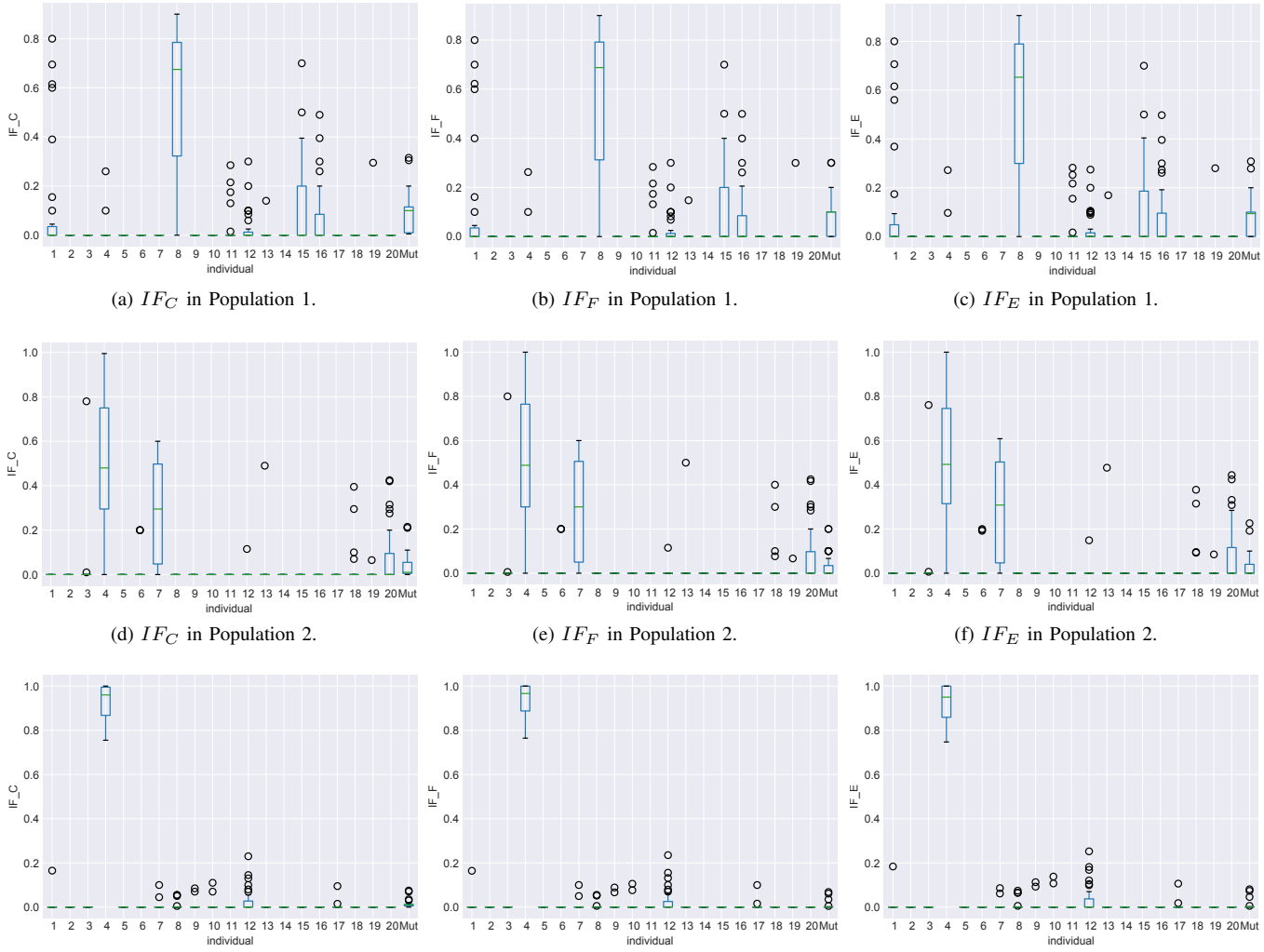
Finally, Population 3 (Figures 9g-9i) got *Individual 5* as the only feasible solution, and so this individual became the most influential in the entire population. *Individual 20* (the second best) was the other one with any influence, and the rest of individuals became mere outliers. It is interesting that the influence of *Individual 5* is tighter bounded in terms of IF_F, and the lower thick becomes lower with IF_E.

For this last problem, as the complexity is larger than the previous problems, it gets more difficult to predict the influence of the individuals. This is due to the constraint handling method. In the constraint handling method, the infeasible individuals located very close to the feasible region do not get a good fitness value when becoming feasible. We believe that other constraint handling methods such as Stochastic Ranking [17] could change this behavior.

V. Conclusion and Future Work

In this paper, we proposed an extension of the general evolutionary algorithm that is able to trace the influence of the individuals from the initial population (and the influence of mutation operations) into successive generations of the algorithm and into the final solutions. For this new algorithm, called T-EA, we proposed three IF metrics to measure the influence of each individual from the initial population: counting-based IF, fitness-based IF and entropy-based IF.

A proof of concept for this framework was performed with three classical problems with different initial populations, illustrating its usability and the usefulness that it can provide for studying the influence of the initial population



(a) IF_C in Population 1. (b) IF_F in Population 1. (c) IF_E in Population 1.

(d) IF_C in Population 2. (e) IF_F in Population 2. (f) IF_E in Population 2.

(g) IF_C in Population 3. (h) IF_F in Population 3. (i) IF_E in Population 3.

Fig. 8: Results obtained for the IF_C, IF_F and IF_E metrics for each initial individual and the mutation operation in the 0/1 knapsack problem with the different initial populations.

and mutation operations into the final solutions. Although in most experiments the three metrics got similar results, it could be observed that in some cases one metric would be more interesting than other in case of measuring the influence in the fitness values or providing a higher impact on influential individuals in "difficult" genes.

Experiments were performed for populations of 20 individuals for the sake of simplicity in the analysis and visualization of results, but it could be tested with larger population sizes (e.g. 100 individuals). This applies for the genome size, which can be easily increased without any further change.

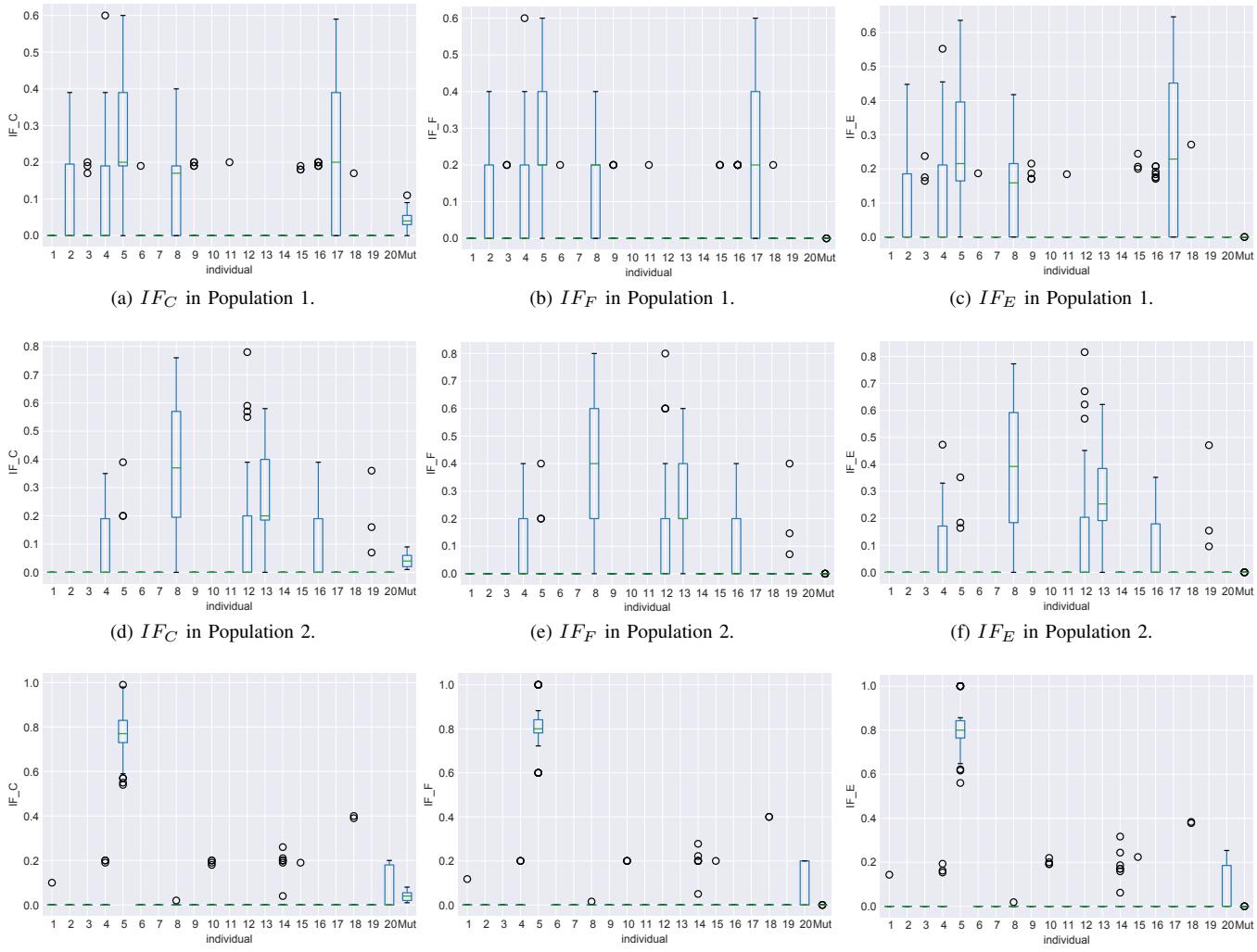
We believe that this framework will be very beneficial when studying new problems and can provide a good insight to better understand the importance of the different components (genes) of the solutions and to decide if seeding the initial population will result beneficial for accelerating the search. For this further study, the impact metrics can be easily extended to study the influence of one individual in the final population for each specific gene of the problem. This will allow to construct a better initial population by giving a higher probability to those alleles that repeated for most final solutions.

In future research works, our aim is to extend this framework to support more representations, such as real-valued vectors and permutations, and also more operators where the inheritance is not always direct from gene to gene, but some genes could be influenced by several parents, such as in arithmetic crossover or polynomial mutation, along with other building blocks that could be interesting to study.

REFERENCES

[1] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "The influence of the number of initial feasible solutions on the performance of an evolu-



(a) IF_C in Population 1. (b) IF_F in Population 1. (c) IF_E in Population 1.

(d) IF_C in Population 2. (e) IF_F in Population 2. (f) IF_E in Population 2.

(g) IF_C in Population 3. (h) IF_F in Population 3. (i) IF_E in Population 3.

Fig. 9: Results obtained for the IF_C, IF_F and IF_E metrics for each initial individual and the mutation operation in the unbounded knapsack problem with the different initial populations.

tionary optimization algorithm," in *Simulated Evolution and Learning*. Springer Berlin Heidelberg, 2012.

[3] S. K. Azad, "Seeding the initial population with feasible solutions in metaheuristic optimization of steel trusses," *Engineering Optimization*, vol. 50, no. 1, pp. 89–105, 2018.

[4] B. İ. Selamoğlu, A. Salhi, and M. Sulaiman, "Strip algorithms as an efficient way to initialise population-based metaheuristics," in *Recent Developments in Metaheuristics*, 2018, pp. 319–331.

[5] H. Maaranen, K. Miettinen, and M. Mäkelä, "Quasi-random initial population for genetic algorithms," *Computers Mathematics with Applications*, vol. 47, no. 12, pp. 1885 – 1895, 2004.

[6] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Computers & Math. with Appl.*, vol. 53, no. 10, pp. 1605 – 1614, 2007.

[7] I. Scriven, A. Lewis, and S. Mostaghim, "Dynamic search initialisation strategies for multi-objective optimisation in peer-to-peer networks," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1515–1522.

[8] M. Kress, S. Mostaghim, H. Schmeck, and D. Seese, "Gap search in particle swarm optimization," in *9th International Conference on Artificial Evolution*, 2009.

[9] D. Sudholt, "How crossover speeds up building block assembly in genetic algorithms," *Evolutionary Computation*, vol. 25, no. 2, pp. 237–274, 2017.

[10] A. V. Eremeev, "On proportions of fit individuals in population of mutation-based evolutionary algorithm with tournament selection," *Evolutionary Computation*, vol. 26, no. 2, pp. 269–297, 2018.

[11] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evo. Comp.*, vol. 10, no. 2, pp. 99–127, 2002.

[12] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[13] J. Schaffer and L. Eshelman, "On Crossover as an Evolutionary Viable Strategy," in *4th International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. Morgan Kaufmann, 1991, pp. 61–68.

[14] S. Sahni, "Approximate algorithms for the 0/1 knapsack problem," *Journal of the ACM (JACM)*, vol. 22, no. 1, pp. 115–124, 1975.

[15] R.-C. Chen and C.-H. Jian, "Solving unbounded knapsack problem using an adaptive genetic algorithm with elitism strategy," in *International Symposium on Parallel and Distributed Processing and Applications*. Springer, 2007, pp. 193–202.

[16] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311 – 338, 2000.

[17] H. Geng, M. Zhang, L. Huang, and X. Wang, "Infeasible elitists and stochastic ranking selection in constrained evolutionary multi-objective optimization," in *Simulated Evolution and Learning*. Springer Berlin Heidelberg, 2006, pp. 336–344.