Multiwinner Voting in Genetic Algorithms for Solving Ill-Posed Global Optimization Problems

Piotr Faliszewski, Jakub Sawicki, Robert Schaefer(✉), and Maciej Smołka

AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
{faliszew,schaefer,smolka}@agh.edu.pl, jsawicki@student.agh.edu.pl

Abstract. Genetic algorithms are a group of powerful tools for solving ill-posed global optimization problems in continuous domains. In case in which the insensitivity of the fitness function is the main obstacle, the most desired feature of a genetic algorithm is its ability to explore plateaus of the fitness function, surrounding its minimizers. In this paper we suggest a way of maintaining diversity of the population in the plateau regions, based on a new approach for the selection based on the theory of multiwinner elections among autonomous agents. The paper delivers a detailed description of the new selection algorithm, computational experiments that guide the choice of the proper multiwinner rule to use, and a preliminary experiment showing the proposed algorithm's effectiveness in exploring a fitness function's plateau.

Keywords: Ill-posed global optimization problems · New tournament-like selection · Fitness insensitivity

1 Introduction

Genetic algorithms (GAs) are a group of powerful tools for solving ill-posed global optimization problems (GOPs) in continuous domains

$$\arg\min_{x\in\mathcal{D}}\{f(x)\},\ \mathcal{D}\subset\mathbb{R}^{\ell},\ f:\mathcal{D}\to\mathbb{R}, \tag{1}$$

where $\mathcal{D}$ is a closed, bounded domain with a nonempty interior and sufficiently regular boundary (e.g., Lipschitz boundary [1]). The ill-conditioning of (1) is frequently due to the existence of many solutions (i.e., due to the multimodality of the objective function f over its domain $\mathcal{D}$) or/and its weak sensitivity in areas surrounding the global or local minimizers (for example, the chart of f may have an almost flat plateau at the level of local minimum $f(\hat{x})$, surrounding the minimizer $\hat{x}$). If the insensitivity is the main obstacle, the genetic algorithm should exhaustively explore plateaus of the fitness function.

The work presented in this paper has been partially supported by Polish NCN grant no. DEC-2015/17/B/ST6/01867 and by the AGH grant no. 11.11.230.124.

G. Squillero and P. Burelli (Eds.): EvoApplications 2016, Part I, LNCS 9597, pp. 409–424, 2016.
DOI: 10.1007/978-3-319-31204-0_27

The taxonomy of managing diversity classifies more than twenty groups of methods (see, e.g., the surveys of Črepinšek [2] and Gupta and Ghafir [3]). The most commonly used are, perhaps, *niching* and *sharing* (see, e.g., Schaefer's book [4] and the work of Goldberg and Richardson [5]). These methods generally lead to populations with sufficient differences in both location and fitness values among the individuals. On the other hand, methods used to increase the efficiency of Multi-Objective Evolutionary Optimization (MOEA) focus on location diversity [6]. There are only a few selection operators designed especially for diversity boosting (see, e.g., the works of Hutter [7] and Matsui [8]).

In this paper we suggest another approach to maintaining diversity of the population in the plateau regions only, based on a new approach for the selection operation. More specifically, we design the selection process for evolutionary algorithms based on the theory of multiwinner elections among autonomous agents. We refer to our approach as the Multiwinner Selection, or MWS for short.

2 Multiwinner Elections

An election is a pair $E = (C, V)$, where $C = \{c_1, \ldots, c_m\}$ is a set of candidates and $V = (v_1, \ldots, v_n)$ is a collection of voters. Each voter v_i has an associated preference order $\succ_i$ that ranks the candidates from the most desirable one to the least desirable one (from the point of view of this voter). For example, if $C = \{a, b, c\}$, then voter v who likes a best, then b, and then c, would have preference order $a \succ b \succ c$.

Given an election $E = (C, V)$, we write $\mathrm{pos}_v(c)$ to denote the position of candidate $c \in C$ in the preference order of voter $v \in V$ (the candidate ranked first has position 1, the next one has position 2, and so on). The exact election that we mean will always be clear from the context.

A multiwinner voting rule is a function $\mathcal{R}$ that given an election $E = (C, V)$ and a positive integer k, $k \leq \|C\|$, outputs a size-k subset of C, the elected committee (ties among winning committees may occur, but we disregard them). So far, multiwinner rules received much less attention from the research community than the single-winner ones. Based on the discussion given by Elkind et al. [9], we consider seven rules inspired by scoring rules. A scoring rule is a function that given a position of a candidate on a voter's preference order returns this candidate's score. For example, k-Approval scoring rule, α_k, is defined so that $\alpha_k(i) = 1$ for $i \in \{1, \ldots, k\}$, and $\alpha_k(i) = 0$ for $i > k$ (α_1 is known as the plurality rule, i.e. the voter approves of a single candidate only). Borda scoring rule—in elections with m candidates—is defined as $\beta(i) = m - i$. Given an election $E = (C, V)$ and scoring function γ, the γ-score of candidate $c \in C$ is defined as $\sum_{v \in V} \gamma(\mathrm{pos}_v(c))$.

Perhaps the easiest way to generalize a scoring rule to the multiwinner case is as follows: Given an election $E = (C, V)$, a scoring protocol α, and the desired number of winners k, simply pick k candidates with the highest scores. This way we define the three following rules:

Single Non-Transferrable Vote (SNTV). Under SNTV we pick the k candidates with the highest plurality scores (i.e., k candidates ranked first most frequently).
***k*-Borda.** Under k-Borda we pick k candidates with the highest Borda scores.
Bloc. Under Bloc we pick the k candidates with the highest k-Approval scores.

The next four rules are based on a somewhat different idea. We first introduce some additional notation and then define the rules of Chamberlin and Courant [10], of Monroe [11], and their approximate versions due to Lu and Boutilier [12] and Skowron et al. [13].

Let $E = (C, V)$ be an election. We say that Φ, $\Phi\colon V \to C$ is a k-CC-assignment function if for each voter $v \in V$, $\Phi(v)$ returns one of at most k candidates (in other words, we require that $\|\Phi(V)\| \leq k$). Intuitively, a k-CC-assignment function matches up to k winners of the election to the voters. Given a k-CC-assignment function Φ and a scoring function α, both for the same election $E = (C, V)$, we say that the score of Φ is:

$$\text{score}_\alpha(\Phi) = \sum_{v \in V} \alpha(\text{pos}_v(\Phi(v))).$$

In other words, each voter v gives score only to the candidate c assigned to him or her.

We now define the Chamberlin–Courant rule (the CC rule). Given an election $E = (C, V)$ and a positive integer k, it picks a k-CC-assignment function Φ with the highest score with respect to the Borda scoring protocol, and returns the committee $\Phi(V)$. Intuitively speaking, the rule picks some k candidates and then assigns each voter to this one of them that this voter ranks highest. The rule picks these k winners in such a way that the sum of the Borda scores that voters give to "their" candidates is highest. In some sense, both the CC-rule and the k-Borda rule are generalizations of the single-winner Borda rule to the multiwinner case. The former, however, divides the electorate into k "districts" of likely-minded voters, and the latter picks k winners that form some sort of a global consensus. Interestingly, the "districts" created by the CC rule can be of very different sizes. In some applications this is undesirable and, thus, Monroe proposed a different variant of this rule.

We say that Φ is a k-Monroe-assignment function if it is a k-CC-assignment function that additionally satisfies the following condition: Let n be the number of voters in the election. For each candidate $c \in C$, it holds that either $\|\Phi^{-1}(c)\| = 0$ or $\lfloor \frac{n}{k} \rfloor \leq \|\Phi^{-1}(c)\| \leq \lceil \frac{n}{k} \rceil$ (in other words, the Monroe condition requires that either a given candidate is not a winner or is a winner and is matched to roughly the same number of voters as the other winners). Monroe rule is the same as the CC rule except that it chooses among k-Monroe-assignments.

While the CC rule and the Monroe rule are quite appealing, it is NP-hard to compute their winners [12,14]. (The situation becomes a bit better if one assumes one of the standard restrictions on the votes such as single-peakedness or single-crossingness; then the CC-rule becomes polynomial-time computable

but the Monroe rule seems to remain intractable [15–17]). Fortunately, there are approximation algorithms for both rules.

Lu and Boutilier [12] introduced a simple greedy algorithm for the CC rule, based on the classic approximation result for submodular functions [18]. This algorithm proceeds as follows. We are given an election $E = (C, V)$ and a positive integer k, the number of winners that we are interested in. We construct the set of winners W. Initially W is empty and we add candidates to it one by one, by executing the following steps k times:

1. For each candidate $c \in C \setminus W$, we compute a k-CC-assignment function Φ_c that assigns each voter v to the candidate in $W \cup \{c\}$ that v ranks highest.
2. We compute for which candidate $c \in C \setminus W$ the score of Φ_c is highest (breaking ties in an arbitrary way).
3. We add this candidate c into W.

We refer to this algorithm as Greedy-CC. Lu and Boutilier have shown that Greedy-CC always picks a committee that under the CC rule would obtain at least fraction $1 - \frac{1}{e}$ of the score of the optimal solution.

For the case of the Monroe rule, there is an approximation algorithm due to Skowron et al. [13]. This algorithm also proceeds greedily, but it makes sure to satisfy the Monroe condition. As for the case of Greedy-CC, this algorithm builds the set of winners by adding candidates to it one by one, but it also maintains the set of available voters. Formally, we have the following algorithm.

Let $E = (C, V)$ be an election and let k be the number of winners that we seek. For the ease of exposition, we assume that k divides $\|V\|$ exactly. The algorithm first sets the set of current winners to be $W = \emptyset$ and the set of available voters to be $A = V$. Then it executes the following steps k times to find the set of k winners and to build a k-Monroe-assignment function Φ:

1. For each candidate $c \in C \setminus W$, compute set
$$A_c = \text{argmax}_{A' \subseteq A, \|A'\| = \frac{\|V\|}{k}} \sum_{v \in A'} \beta(\text{pos}_v(c))$$
(break ties arbitrarily, if needed; intuitively, A_c is a set of $\frac{\|V\|}{k}$ voters from A that jointly rank c highest). For each c let score(c) be $\sum_{v \in A_c} \beta(\text{pos}_v(c))$.
2. Pick candidate $c \in C \setminus W$ with the highest score (breaking ties arbitrarily, if needed). Add c to the set W, remove the voters from A_c from A, and for each voter v in A_c set $\Phi(v) = c$.

Finally, the algorithm returns the set W. We refer to the algorithm as Greedy-Monroe and treat it as a voting rule in its own right. Skowron et al. have shown that the score of the k-Monroe-assignment function Φ returned by Greedy-Monroe is at least $n(m-1)\left(1 - \frac{k-1}{2(m-1)} - \frac{H_k}{k}\right)$, where n is the number of voters, m is the number of candidates, and H_k is the k'th harmonic number (i.e., $H_k = \sum_{i=1}^{k} \frac{1}{i}$). For the case where k is relatively large and $\frac{k}{m}$ is relatively small, this value is very close to the highest possible score under the Monroe rule, $n(m-1)$, achieved by an assignment function that matches every voter with his or her most preferred candidate.

3 Selection Based on Multiwinner Voting

The input for the selection procedure consists of election group C, which is a subset of population $X_t = \langle x^{(1)}, \ldots, x^{(\mu)} \rangle$, the multiset of candidate solutions at the particular t-th epoch of evolutionary optimization process (each $x^{(i)}$ is a point in an Euclidean space). As long as it does not lead to ambiguity, we do not specify the dependency of each individual in X_t on the epoch number. For every point $x^{(i)} \in C$, we have its fitness value $f(x^{(i)})$, the smaller the fitness value the better (since in the multiwinner voting we maximize rather than minimize, we will have to apply appropriate transformations of these values). We are to pick k points from C that will be the parents in the following mixing phase. (Typically, a selection procedure is applied multiple times, each invocation producing a single individual. In our case we will still invoke the selection procedure several times, but each invocation will output a collection of individuals; we formalize this in Sect. 3.4).

Our idea is to consider C as a group of individuals who need to decide which k of them would survive to the next epoch. These individuals are driven by two, perhaps conflicting, desires.

1. Foremost, each individual would like to survive itself. If, however, the individual were not to survive, it would like some as similar as possible individual to survive. Intuitively, a similar individual would have similar genes that would be passed to the next generation.
2. The second desire is that the selected subpopulation is as fit as possible.

To model these desires, we introduce for each individual $x^{(i)}$ its utility function $u_i \colon C \to \mathbb{R}$ (we will provide some examples of utility functions very shortly). For each two individuals $x^{(i)}$ and $x^{(j)}$, the value $u_i(x^{(j)})$ expresses how much value $x^{(i)}$ attaches to $x^{(j)}$ being selected (the higher the value, the more $x^{(i)}$ would like $x^{(j)}$ to be chosen). Naturally, given the principles outlined above, for each two individuals $x^{(i)}, x^{(j)}$, we have that

$$u_i(x^{(i)}) \geq u_i(x^{(j)}).$$

That is, foremost each individual is selfish and its desire to survive is stronger than anything else.

Given the set of individuals and their utility functions, we define an election $E = (C, V)$ as follows:

1. The set of voters V is the same as the set of candidates (the individuals), that is, $V = C$.
2. For each individual $x^{(l)}$, we set its preference order so that if $u_l(x^i) > u_l(x^{(j)})$ then $x^{(l)}$ prefers $x^{(l)}$ to $x^{(j)}$. (We break the ties arbitrarily, should they occur.)

Now, given election E, we simply apply a multiwinner voting rule of choice to pick k winners.

3.1 Utility Functions

There are two crucial choices in the design of MWS. The choice of the multiwinner voting rule and the choice of the utility function. Here we outline several possibilities for the latter.

Perhaps the most natural idea is to use the following approach. For each two individuals $x^{(i)}$ and $x^{(j)}$, let $d(x^{(i)}, x^{(j)})$ be a distance between them (one could use any metric, but for simplicity we use the Euclidean distance). We also assume to have "reversal" function h such that for each two individuals $x^{(i)}$ and $x^{(j)}$ it holds that $f(x^{(i)}) \leq f(x^{(j)})$ if and only if $h(f(x^{(i)})) \geq h(f(x^{(j)}))$. Then we define the utility of individual $x^{(j)}$ from the point of view of individual $x^{(i)}$ to be:

$$u_i^p(x^{(j)}) = \frac{h(f(x^{(j)}))}{d(x^{(i)}, x^{(j)})}.$$

We refer to these functions as proportional utilities because they are directly proportional to the "reversed" fitness values and inversely proportional to the distances between the individuals (hence the symbol p in u_i^p).

It is easy to see that proportional utilities satisfy the basic desiderata outlined in the above section. Since they are inversely proportional to the distance between the individuals, each individual assigns the highest utility ($+\infty$) to itself and decreases its utility with increasing the distance (with increasing the dissimilarity) to the other individuals. Since the utilities are proportional to the reversed fitness values, the individuals assign value selecting individuals as fit as possible.

Naturally, it might be the case that the proportional utilities either put too much stress on the fitness values or too much stress on the distances between the agents. Thus, to temper this behavior, we might need to use the following variant of proportional utilities. Let γ and δ be two functions $(\gamma, \delta \colon \mathbb{R} \to \mathbb{R})$, where γ is increasing and δ is decreasing. We define (γ, δ)-proportional utility function to be:

$$u_i^{\gamma,\delta}(x^{(j)}) = \gamma(h(f(x^{(j)}))) \cdot \delta(d(x^{(i)}, x^{(j)}))$$

For example, by taking $\gamma(x) = x$ and $\delta(x) = \frac{1}{x}$ we obtain the proportional utility functions.

The choice of functions γ and δ is likely to have very strong impact on the quality of our selection procedure. Indeed, we believe that exploring various functions may be an interesting research project in its own right. In this work we will focus on one particularly appealing type of (γ, δ)-proportional utilities, where γ and δ are of the following form: For two positive numbers r and s, we take $\gamma_r(x) = x^r$ and $\delta_s(x) = x^{-s}$. Naturally, even in this case, choosing appropriate values of r and s is not obvious.

3.2 The Selection Procedure and the Choice of the Multiwinner Rule

The pseudocode of our selection procedure is given as Algorithm 1. This code is quite general and can use any (γ, δ)-utilities. It would also be straightforward

Algorithm 1. The Multiwinner Selection (MWS) procedure. The goal is to pick k individuals from the election group C, based on their locations and fitness values, using multiwinner voting rule $\mathcal{R}$.

Notation:
$C = \langle x^{(1)}, \ldots, x^{(n)} \rangle \subseteq X_t \leftarrow$ the election group
$f \leftarrow$ the fitness function
$h \leftarrow$ the "reversal" function
$d \leftarrow$ the metric over $\mathbb{R}^\ell$
$\gamma, \delta \leftarrow$ the functions defining (γ, δ)-proportional utilities
$k \leftarrow$ the number of individuals to pick
$\mathcal{R} \leftarrow$ the multiwinner voting rule

// prepare the election among the individuals
$V \leftarrow (v_1, \ldots, v_n)$
for $l \leftarrow 1$ **to** n **do**
  foreach $i, j \in \{1, \ldots, n\}$, $i < j$ **do**
    $u_l(x^{(i)}) \leftarrow \gamma(h(f(x^{(i)})) \cdot \delta(d(x^{(l)}, x^{(i)}))$
    $u_l(x^{(j)}) \leftarrow \gamma(h(f(x^{(j)})) \cdot \delta(d(x^{(l)}, x^{(j)}))$
    if $u_l(x^{(i)}) > u_l(x^{(j)})$ **then**
      set v_l's preference order so that $x^{(i)} \succ_l x^{(j)}$
    else
      set v_l's preference order so that $x^{(j)} \succ_l x^{(i)}$
form election $E = (C, V)$
$W = \mathcal{R}(E, k)$; // hold the virtual election among the individuals
output W

to adapt it to any other natural form of utilities. The code can also use any arbitrary multiwinner voting rule $\mathcal{R}$. However, it should be quite clear that the quality of the procedure will deeply depend on the choice of the rule. Indeed, if we used SNTV which simply counts how many times each candidate is ranked first, our procedure would—in essence—reduce to randomly selecting a group of k individuals. This is so because each individual ranks itself first and, thus, every individual would simply have one point.

Similarly, we believe that k-Borda and Bloc would not perform very well. The single-winner variant of k-Borda is designed to find a "consensus" winner, that is, a candidate that is in some sense acceptable to as many voters as possible. In effect, k-Borda finds a collection of such "consensus" candidates and they are likely to be very similar. Bloc rule might be a bit better because it is based on the k-Approval scoring protocol and, effectively, under Bloc the score of each candidate depends on local information only. This might, however, lead to picking individuals from large clusters only.

On the other hand, we believe that the CC rule and the Monroe rule would not suffer from the above-described problems. The reason is that under these rules, when a candidate is assigned to some voter, this voter cannot contribute

to the score of any other candidate. As a result, members of a big cluster would only be able to promote *some* of their number into the winning set, preventing overwhelming of smaller clusters.

For the same reason, it seems that Greedy-CC and Greedy-Monroe should perform well, and should be faster than CC and Monroe (since computing CC and Monroe requires solving NP-hard problems). Indeed, using Monroe and CC for any non-trivial setting seems impossible. On the other hand, Greedy-CC and Greedy-Monroe are computed through simple, polynomial-time algorithms and, in effect, the computational cost of using multiwinner selection based on them is negligible (as compared to the cost of fitness evaluations for engineering applications for which our techniques are intended). We also note that using multiwinner selection requires the same number of fitness evaluations as, say, tournament selection.

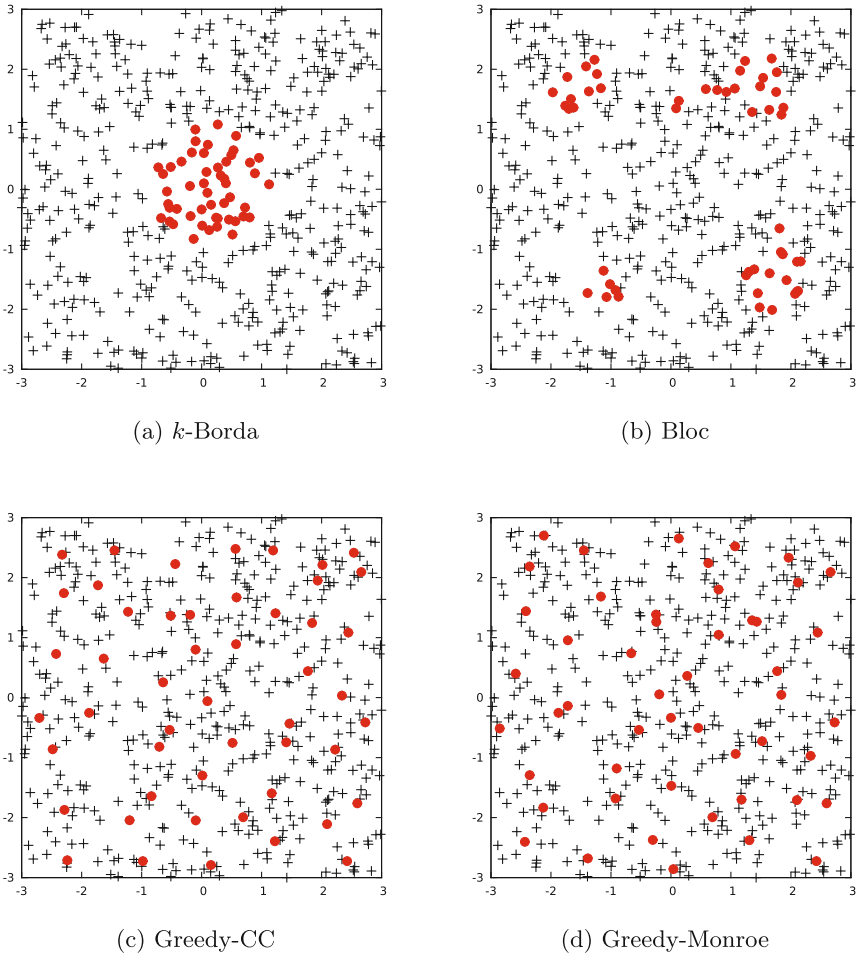
3.3 A Simple Experiment

The main reason for developing our Multiwinner Selection procedure is to enhance the plateau exploration capabilities of genetic algorithms. In the preceding section we have argued theoretically that to achieve this effect we should, likely, use Greedy-CC or Greedy-Monroe rules. However, instead of relying on theoretical analysis only, we believe that it would be informative to perform a simple computational experiment that would give us some further insight into the behavior of MWS depending on the applied voting rule.

To this end, we have performed the following experiment. We have generated 500 points distributed uniformly on the two-dimensional square $[-3, 3] \times [-3, 3]$ and have applied our selection procedure to pick 50 points. We have assumed that each point has the same fitness value (thus the choice of function h is irrelevant) and we have used the unmodified 2-dimensional Euclidean distance (that is, we have used $\delta(x) = \frac{1}{x}$). This setting models the situation in which the genetic algorithm has hit the plateau and now the goal is to explore it. We would like to obtain as much diversity among the selected individuals as possible.[1]

The results of the experiment, presented in Fig. 1, are quite striking and fully support the theoretical discussion from the preceding section: k-Borda picks a centrally located cluster of individuals, Bloc picks candidates from areas where they are concentrated (due to the random choice of their positions), whereas Greedy-CC and Greedy-Monroe pick candidates that are, approximately, uniformly distributed among the individuals. In effect, we believe that k-Borda and Bloc would be poor choices for our application. Greedy-CC and Greedy-Monroe would, likely, perform comparatively well. However, Greedy-CC is faster to compute and since it does not have to respect the Monroe criterion, it is less likely to

[1] From the point of view of the elections theory, our setting is an example of two-dimensional Euclidean single-peaked preferences. Under two-dimensional Euclidean preferences, every voter and every candidate is a point in a two-dimensional Euclidean space and every voter (in our case, every individual) derives his or her preference orders by sorting the candidates (in our case, the individuals) with respect to their Euclidean distance from him or herself.



(a) k-Borda (b) Bloc

(c) Greedy-CC (d) Greedy-Monroe

Fig. 1. The result of using Multiwinner Selection to pick 50 out of 500 individuals, distributed uniformly on a $[-3, 3] \times [-3, 3]$ square, using k-Borda, Bloc, Greedy-CC, and Greedy-Monroe. All the individuals have the same fitness value, simulating the plateau scenario (in effect, the utilities, and the preference orders, are derived based on the distance between points only). For the case of each rule we use the same 500 points.

pick two very similar points (this would happen for the case of Greedy-Monroe if these two points were surrounded by a denser-than-usual cluster of points). Thus, from now on in MWS we will use Greedy-CC only.

We have shown a result of a single invocation of our experiment, but we have run it repeatedly and each time we obtained similar results. Since for now we are interested in qualitative results rather than quantitative ones, we believe the obtained evidence is sufficient to focus on Greedy-CC only.

Algorithm 2. Outline of a genetic algorithm using MWS.

Notation:
$X_t = \langle x_t^{(1)}, \ldots, x_t^{(\mu)} \rangle \in \mathcal{D}^\ell \leftarrow$ the population in the t-th epoch
$C \subseteq X_t \leftarrow$ the election group
$n \leftarrow$ the size of the election group ($n \leq \mu$)
$k = 2p \leftarrow$ the number of individuals to pick
MWS $\leftarrow$ Multiwinner Selection procedure

```
1  Sample the initial population X_0
2  Evaluate X_0
3  t ← 0
   // the main loop over the epochs
4  while Stopping_Condition(X_t) do
5  |  Offspring ← ∅
6  |  for i ← 1 to μ/p do
7  |  |  Choose the election group C of size n from X_t
   |  |  // Select k individuals from the election group
8  |  |  {c_1, ..., c_k} ← MWS(C, k)
9  |  |  for j ← 1 to p do
10 |  |  |  r ← 2j − 1
11 |  |  |  a ← cross(c_r, c_{r+1})
12 |  |  |  a ← mutate(a)
13 |  |  |  Offspring ← Offspring ∪ {a}
14 |  t ← t + 1
15 |  X_t ← Offspring
16 |  Evaluate X_t
17 output X_t
```

3.4 Genetic Algorithm Using Multiwinner Selection

Let μ be the number of individuals in the population in a single epoch. Further, we choose two numbers, n and p, where n is the size of the election group (i.e., the number of individuals over which we will carry a multiwinner election), and p, $p < \mu$, is a number that divides μ and such that we pick $k = 2p$ individuals from the election group.

A single iteration of our algorithm proceeds as follows. We pick the election group, that is, a set $C \subseteq X_t$ of individuals of size n, where X_t is the current population. (There are at least several ways of choosing X_t and we will discuss two possibilities shortly; for now let us simply think of it as some stochastic process that picks n individuals). Then we apply Multiwinner Selection procedure to pick k individuals out of these n. Finally, we iterate over these k individuals, pick consecutive pairs, apply the crossing operation to the individuals from the pair, obtaining a single individual, and finally we apply the mutation operator to this individual (we assume that the crossing operator and the mutation operator encapsulate the stochastic choices as to whether they should, all in all, be applied

or not). In effect, we obtain a group of p individuals that we insert into the next epoch's population. We repeat this process μ/p times, to form a population of size μ.

Let us now move back to the issue of picking the election group. We suggest two ways in which this can be done:

1. We obtain the election group C by sampling without replacement n times from the current population X_t (using the uniform distribution). This approach is inherited from the standard tournament selection procedure.
2. The second way is composed of four steps. In the first step we select a single individual x_{seed} from X_t using some conventional selection procedure (e.g., the proportional one or the tournament one). In the second step we chose the normal, ℓ-dimensional distribution with a density function ρ, whose expectation is x_{seed}. The standard deviation σ of this distribution is a parameter of the procedure. Next, we create the probability distribution χ on the multiset $X_t \setminus \{x_{seed}\}$ by normalizing the vector $\{\rho(x^{(i)})\}$, $x^{(i)} \in X_t$, $i = 1, \ldots \mu$, $x^{(i)} \neq x_{seed}$. Finally, we obtain the election group C by $(n-1)$-times sampling without replacement from $X_t \setminus \{x_{seed}\}$, according to the probability distribution χ. Finally, we add x_{seed} to the election group.

From now on, we will focus only on the first, far simpler, way of picking the election group. However, naturally, it has drawbacks. For example, the election group picked in this way might be too diverse in case of a relatively large search domain. We believe that the second procedure for picking the election group would resolve this problem. Measuring the extent to which this problem indeed occurs in practice is beyond the scope of this paper.

4 Experimental Evaluation

In this section we provide evidence that our Multiwinner Selection method indeed achieves its goal, i.e., it leads to the exploration of plateau areas (without necessary focusing on a single local or global optimum). To this end we consider the following experiment. Let $f(x, y)$ be a function, $f\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, defined as:

$$f(x,y) = \max\left(\frac{1}{2} - e^{-2(x^2+y^2)}, 0\right) + \frac{20 + x^2 - 10\cos(2\pi x) + y^2 - 10\cos(2\pi y)}{2000}$$

While $f(x, y)$ might look somewhat complicated at first, its definition is in fact very simple. The first term of $f(x, y)$, $\max(\frac{1}{2} - e^{-2(x^2+y^2)}, 0)$ is a simple Gaussian function reversed, translated up to $\frac{1}{2}$, and cut off at 0. In effect, this part of the function creates a well, with a plateau in the shape of a circle with the center at point $(0, 0)$ and radius $r = \sqrt{\frac{-\ln\frac{1}{2}}{2}} \approx 0.58$. The second summand is the Rastrigin function (downscaled by a factor of 2000) to introduce small perturbations. We plot the function in Fig. 2.

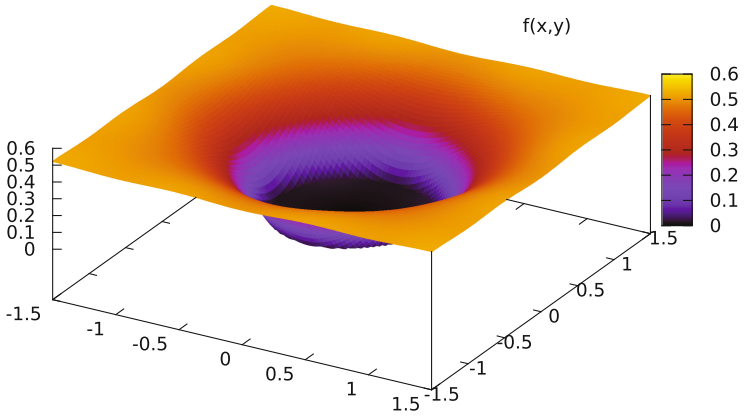


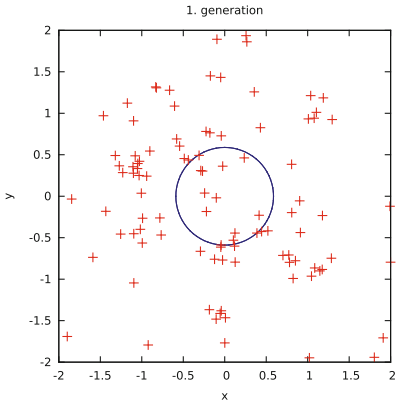
Fig. 2. The plot of the function $f(x, y)$. It is, in essence, a well with the center $(0, 0)$; the small perturbations due to the Rastrigin function are barely visible.

The idea of our experiment is to run a genetic algorithm whose goal is to find minimizers of function $f(x, y)$ (on the domain $[-2, 2] \times [-2, 2]$). Naturally, a standard algorithm would very quickly find the global minimum at point $(0, 0)$. However, what we are interested in is not finding the global minimum, but exploring the plateau area in the circle of radius $r \approx 0.58$, centered at $(0, 0)$. Arguably, all the points in this area are of very similar quality, and from our point of view it is important to cover as much of this area as possible.

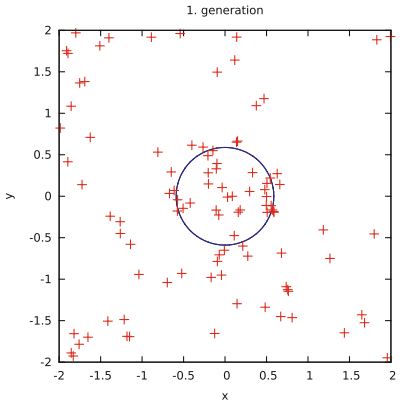
To this end, we compare two algorithms. Our algorithm from the previous section, using the Multiwinner Selection procedure, and a simple standard genetic algorithm using a form of the tournament selection procedure. Both algorithms use the following basic parameters:

1. The population in each epoch contains $\mu = 100$ individuals.
2. The initial population is picked by drawing μ points from $[-2, 2] \times [2, 2]$ uniformly at random.
3. The probability of performing the crossover operation is 1 %, whereas the probability of mutation is 100 %. Mutation is executed by adding to the individual a value drawn randomly using the normal distribution with standard deviation 0.1.

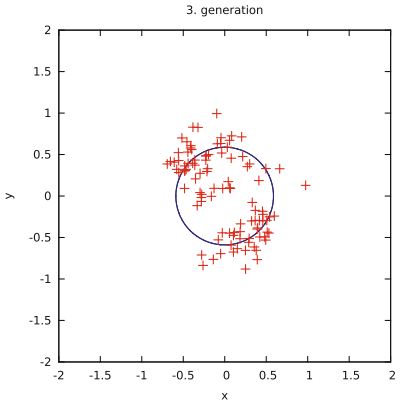
Arguably, this setting of the parameters (especially the mutation rate and the standard deviation of the normal distribution used for mutation) is quite extreme. However, what we are modeling here is a situation where the algorithm already, roughly, identified the part of the domain with a plateau (thus we look at the domain $[-2, 2] \times [-2, 2]$) and now the goal is to fill in this plateau. For this task we want the population to be exploration-oriented and, thus, we want the mutation operator to have strong effect, and we want the crossover operation to have very small impact. Thus, We use the following parameters:



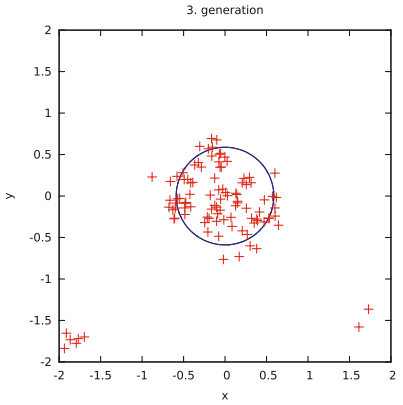
(a) Epoch 1 (Tournament)



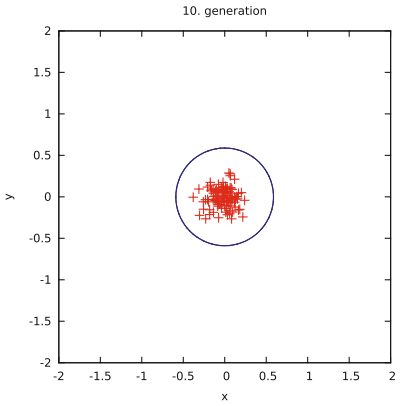
(b) Epoch 1 (Multiwinner)



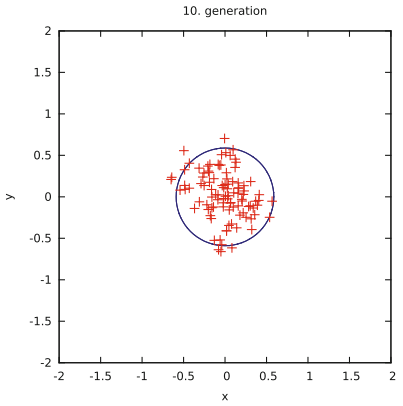
(c) Epoch 3 (Tournament)



(d) Epoch 3 (Multiwinner)



(e) Epoch 10 (Tournament)



(f) Epoch 10 (Multiwinner)

Fig. 3. Sample run of our algorithms with tournament and Multiwinner selection procedures (populations X_1, X_3, and X_{10}). The blue circle depicts the plateau region.

1. For the Multiwinner Selection, we use (γ, δ)-proportional utilities, with $\gamma(x) = x^6$ and $\delta(x) = \frac{1}{x}$. We use the "reversal" function $h(x) = \frac{1}{1+x}$. Further, we set $n = 30$ and $\mu = 5$ (that is, every selection process consists of holding $\frac{100}{5} = 20$ elections among randomly chosen 30 individuals, of whom, eventually, 5 are selected as the parental individuals).
2. For the tournament selection, we have used the following process: To pick each parental individual we pick 5 randomly selected individuals and then draw one of them with probability proportional to their fitness value (the better the fitness value, the better the chance of being selected).

We should mention that the above parameter setting is largely ad-hoc. We did not try to optimize the values and we have performed only a handful of preliminary experiments to asses the behavior of the system. This is in sync with the fact that what we present here is nothing more than a proof of concept that Multiwinner Selection is a tool worth developing in the context of extending the capabilities of plateau exploration for genetic algorithms.

We have run our algorithms on function $f(x, y)$ for 20 times each, in each case executing 10 epochs (i.e., starting with population X_0, and computing populations $X_1, \ldots, X_{10}$). In Fig. 3 we present populations X_1, X_3 and X_{10} for both algorithms, from representative runs. This figure shows that our intuition that Multiwinner Selection should lead to better exploration of the plateau region is correct. However, we also assessed the extent of this advantage quantitatively.

To this end, we have computed how much of the plateau region is covered by the individuals from each of the populations. The plateau region is a circle with the radius $r \approx 0.58$. Since mutation in our algorithm modifies the position of

Table 1. Comparison of the average fraction of the plateau covered by the algorithm using tournament selection and Multiwinner selection. For each algorithm we report the average fraction of the plateau covered by populations $X_1, \ldots, X_{10}$ and its standard deviation. (The average and the standard deviation is computed over the 20 test runs that we have executed.)

	covered fraction of the plateau	
population	average	st. dev.
X_1	0.1124	0.0431
X_2	0.2165	0.0712
X_3	0.3521	0.0696
X_4	0.3846	0.0400
X_5	0.3586	0.0554
X_6	0.3118	0.0545
X_7	0.2765	0.0455
X_8	0.2588	0.0305
X_9	0.2474	0.0152
X_{10}	0.2428	0.0120

(a) Tournament Selection

	covered fraction of the plateau	
population	average	st. dev.
X_1	0.1152	0.0385
X_2	0.2511	0.0634
X_3	0.3613	0.0454
X_4	0.4011	0.0278
X_5	0.4202	0.0255
X_6	0.4139	0.0261
X_7	0.4251	0.0195
X_8	0.4276	0.0262
X_9	0.4207	0.0175
X_{10}	0.4234	0.0161

(b) Multiwinner Selection

each individual according to a two-dimensional normal distribution with standard deviation 0.1, we have assumed that each individual covers an area of the plateau equal to the circle of radius 0.05, centered at the individual. Then, we have computed the fraction of the area of the plateau covered by at least one individual. We have done so for both algorithms, for every run of our algorithm, and for every epoch within the run. We present the results in Table 1.

The results confirm our intuition. The algorithm using Multiwinner selection achieves a better coverage (nearly twice as large as that using tournament selection) and does not converge to the local optimum. Instead, it maintains the diversity among the individuals. Since the standard deviation for the covered fraction of plateau in the tenth epoch is very small (for both algorithms), this is good indication that our results are meaningful.

We should mention that the results from Table 1 are very sensitive to our definition of what it means for an individual to "cover a given area of the plateau." For example, it depends on the radius "covered" by a single individual. However the main message, that Multiwinner Selection ensures noticeably better diversity of the individuals than tournament selection remains true.

5 Conclusions

We have put forward a new idea for a selection procedure for genetic algorithms. This selection procedure is based on the theory of multiwinner voting, and its goal is to maintain diversity of the population within plateau regions. We have considered several voting rules that can form the basis of the selection procedure and we have concluded that a greedy approximation algorithm for Chamberlin–Courant's rule is promising (however, the approximation algorithm for Monroe's rule is promising as well). Then we have tested our selection procedure within a very simple genetic algorithm. We have applied the algorithm to a toy example of a function with a plateau. We have shown that compared to the same algorithm with tournament selection, our algorithm achieves noticeably better diversity of its populations. This is seen both by a quantitative assessment and by inspecting the populations visually.

The research presented in this paper is very preliminary. It is very likely that there are better ways of employing our ideas, there are better parameter settings, etc. Nonetheless, we believe that our results are sufficient to support our main message: Selection based on the theory of multiwinner voting is an interesting idea that can lead to diversity among the individuals in genetic algorithms.

References

1. Zeidler, E.: Nonlinear Functional Analysis and its Application: II/A: Linear Monotone Operators. Springer, New York (2000)
2. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput. Surv. **45**(3), 3–35 (2013)

3. Gupta, D., Ghafir, S.: An overview of methods maintaining diversity in genetic algorithms. Int. J. Emerg. Technol. Adv. Eng. **2**(5), 56–60 (2012)
4. Schaefer, R.: Foundation of Genetic Global Optimization (with Chap. 6 by Telega H.). Studies in Computational Intelligence Series, vol. 74. Springer, Heidelberg (2007)
5. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. genetic algorithms and their applications. In: Proceedings of 2nd International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, pp. 41–49 (1987)
6. Bosman, P., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(2), 174–188 (2003)
7. Hutter, M.: Fitness uniform selection to preserve genetic diversity. In: Proceedings of the 2002 Congres of Evolutionary Computation, pp. 783–788 (2002)
8. Matsui, K.: New selection method to improve the population diversity in genetic algorithms. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 625–630 (1999)
9. Elkind, E., Faliszewski, P., Skowron, P., Slinko, A.: Properties of multiwinner voting rules. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, pp. 53–60, May 2014
10. Chamberlin, B., Courant, P.: Representative deliberations and representative decisions: proportional representation and the Borda rule. Am. Polit. Sci. Rev. **77**(3), 718–733 (1983)
11. Monroe, B.: Fully proportional representation. Am. Polit. Sci. Rev. **89**(4), 925–940 (1995)
12. Lu, T., Boutilier, C.: Budgeted social choice: from consensus to personalized decision making. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 280–286 (2011)
13. Skowron, P., Faliszewski, P., Slinko, A.: Fully proportional representation as resource allocation: approximability results. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, pp. 353–359. AAAI Press (2013)
14. Procaccia, A., Rosenschein, J., Zohar, A.: On the complexity of achieving proportional representation. Soc. Choice Welfare **30**(3), 353–362 (2008)
15. Betzler, N., Slinko, A., Uhlmann, J.: On the computation of fully proportional representation. J. Artif. Intell. Res. **47**, 475–519 (2013)
16. Skowron, P., Yu, L., Faliszewski, P., Elkind, E.: The complexity of fully proportional representation for single-crossing electorates. In: Vöcking, B. (ed.) SAGT 2013. LNCS, vol. 8146, pp. 1–12. Springer, Heidelberg (2013)
17. Elkind, E., Faliszewski, P., Skowron, P.: A characterization of the single-peaked single-crossing domain. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, pp. 654–660 (2014)
18. Nemhauser, G., Wolsey, L., Fisher, M.: An analysis of approximations for maximizing submodular set functions. Math. Program. **14**(1), 265–294 (1978)