Journal of Computational Science 80 (2024) 102328

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Voting according to one's political stances is difficult: Problems definition, computational hardness, and approximate solutions☆

Aitor Godoy [a], Ismael Rodríguez [a,b], Fernando Rubio [a,b,*]

[a] Dept. Sistemas Informáticos y Computación. Facultad de Informática Universidad Complutense de Madrid, 28040 Madrid, Spain
[b] Instituto de Tecnología del Conocimiento. Universidad Complutense de Madrid, 28040 Madrid, Spain

ARTICLE INFO

Keywords:
Computational complexity
Approximability
Polynomial reductions
NP-complete problems
Political problems
Electoral systems

ABSTRACT

This paper studies the computational complexity of two voting problems where the goal is deciding how a given voter should vote to favour their personal stances. In the first problem, given (a) the voter stance towards each law that will be voted by the parliament and (b) the political stance of each party towards each law (all party members are assumed to vote according to it), the goal is finding the parliamentary seats distribution maximizing the number of laws that will be approved/rejected as desired by the voter. In the second problem no parliament is involved, but a single issue with several possible answers is voted by citizens in a presidential election with several candidates. The problem consists in deciding how a group of voters, split in different electoral districts, all of them supporting the same candidate, should vote to make their candidate president. It is assumed that (a) all delegates of each electoral district are assigned to the candidate winning in the district, (b) after the election day, candidates may ask their assigned delegates to support other candidates receiving more votes than them, and these post-electoral supporting stances are known in advance by the electorate, and (c) the group of voters that is coordinated knows the votes that will be cast by the rest of the electorate. For each problem, its NP-hardness as well as its inapproximability are proved. This implies that something as essential as exercising the democratic right to vote, in such a way that the voting choice will be the best for the voter's political stances, is at least NP-hard. It is also shown how genetic algorithms can be used to obtain reasonable solutions in practice despite the limitations of theoretical approximation hardness.

1. Introduction

At a first glance, voting in an election may seem to be a relatively simple action from the voter's perspective. Apparently, all a voter has to do is to vote for the political party whose viewpoint aligns most with their own. Yet it is likely that, in some election, the voter had to vote for a party that was not the best compared to others in terms of the laws they wanted to pass, because of the pacts that other political parties were expected to make within them. In other words, voters have to make some strategic analysis in order to decide what to vote, regardless of which political option they may like best.

In this paper the (NP-)hardness of two problems involving the individual choice of how to vote is established. In them, the difficulty of voting will not lie in deciding which party a voter agrees more with, but in how individuals decide what to vote to favour their goals. It will be assumed that politicians are fully consistent with their claims and pre-electoral voting surveys are perfect. The proposed properties will show that, even under this particularly favourable setting without uncertainty, choosing what to vote is NP-hard.

The mathematical properties of different electoral systems have been studied in many previous works (see e.g. [1,2]). For instance, very detailed studies have been carried out on how to do *gerrymandering*, i.e. how to design constituencies to favour particular parties or candidates (see e.g. [3,4]), including studies about the computational complexity of gerrymandering (see [5–7]). In the problems under study in this paper, the goal will be searching for the best decision for particular *voters* in two fixed electoral models based on real-world elections. These problems will be in a simple parliamentary election model and a simple presidential election one, both designed to capture the essence of the mechanics of typical elections in many democratic countries.

☆ Work partially supported by projects PID2019-108528RB-C22, and by Comunidad de Madrid as part of the program S2018/TCS-4339 (BLOQUES-CM) co-funded by EIE Funds of the European Union.

* Corresponding author at: Dept. Sistemas Informáticos y Computación. Facultad de Informática Universidad Complutense de Madrid, 28040 Madrid, Spain.
E-mail addresses: aitorgod@ucm.es (A. Godoy), isrodrig@ucm.es (I. Rodríguez), fernando@sip.ucm.es (F. Rubio).

https://doi.org/10.1016/j.jocs.2024.102328
Received 7 October 2022; Received in revised form 13 March 2024; Accepted 16 May 2024
Available online 18 May 2024

1.1. Problems under study

In the first problem, a parliamentary election is considered, and the goal is to vote in such a way that the laws the resulting parliament will pass are most aligned with the voter's own political ideals.[1] In this context, the question is whether there is a configuration of parliament that would allow all (or the greatest number of) target laws to be passed, given the stance (in favour, against or neutral) of each party towards each law. That is, is there a way to assign the seats of the parliament in such a way that some given number of the desired laws are passed when voted by the parliament? The NP-completeness of this problem will be shown. Moreover, another result will also show that it is even hard to guarantee a good approximation ratio in this problem.

The second problem considers a presidential election in an electoral system divided into constituencies with an all-or-nothing delegates distribution, i.e. the winner in each constituency wins all the delegate votes to choose president assigned to the constituency. It is known that each candidate will use their gained delegates to support the candidate most voted among all other like-minded candidates if they are not the most voted one. Some voters supporting the same candidate are split into different constituencies, and the objective of the problem is to check out whether these voters can coordinate their votes so that their preferred candidate wins the election. It will be shown that this problem is NP-complete, and that the problem of maximizing the number of representatives of the target candidate cannot be polynomial-time approximated at any level if $\mathbf{P} \neq \mathbf{NP}$.

Regarding the first problem, to the best of our knowledge the effect of the "proxy" decision-making induced by parliamentary elections on the computational complexity of voting has not been studied before.[2] On the other hand, the second problem can be seen as a kind of *manipulation problem*, which are problems where a set of voters try to coalesce to manipulate the outcome of an election (see e.g. [8–11]). However, it is neither a generalization nor a particularization of any previous manipulation problem in the literature as far as we know. Seeing the problem according to the notation in [9], the adopted scoring rule is *Plurality* (i.e. a single voting point is given by each voter to their preferred choice, and no other point is given to any other), although the problem is quite different to the so-called *Plurality with runoff* problem because there is no second voting round asking for any additional preference information from the voters. In fact, the introduction of *constituencies* in the problem proposed in this paper will be key for its difficulty, yielding its NP-hardness.

After the computational hardness of both problems is established, experiments are conducted where the problems are (sub-)optimally solved. Genetic Algorithms (see e.g. [12,13]) are chosen to find reasonable sub-optimal solutions for both problems under consideration. Experimental results showing the usefulness of these algorithms are reported.

The main contributions of this work are the following:

(a) formally introducing two specific problems involving the voter's strategic choice which are designed to resemble real-world elections more closely—in particular, by complicating the choice with realistic factors such as the perturbing effects of constituencies and alliances, and the difficulty of voting a parliament rather than the laws themselves;
(b) determining both the computational complexity and the approximation hardness of such problems;
(c) providing heuristic solutions for them by means of genetic algorithms.

The rest of the paper is organized as follows. First, the problems are formally defined in Section 2. The NP-completeness and the polynomial-time inapproximability of each problem are proved in Section 3. Afterwards, in Section 4 a genetic algorithm is presented for each problem, and experimental results are reported for several problem instances. In Section 5 the main findings of the paper are discussed, and the final conclusions and lines of future work are presented in Section 6. Let us point out that proving the inapproximability of the problems also implies their NP-hardness. Nevertheless, we prefer to show both proofs, instead of presenting only the inapproximability proofs, to simplify the understanding of the steps followed in our reasoning.

2. Formal definition of the problems

The problems under consideration, called PARLIAMENT and PRESIDENT, are formally introduced in this section. Their decision and optimization versions are defined, and a detailed explanation of each problem is presented.

2.1. PARLIAMENT problem

Given which political parties are in favour of, neutral, or against some laws, the voter's goal is forming a parliamentary seats distribution that will pass the largest number of laws aligning with their own political stance.

It is assumed that the voter knows in advance which parties will vote for each of these laws. Note that, although a voter can be against law L, we simplify the problem by replacing this law by its opposite (not L), thereby allowing us to assume that the voter is in favour of all laws under consideration. Hence, for the sake of simplicity, it is assumed that our voter is in favour of all laws under consideration.[3] Also, note that if the voter's personal stance is *neutral* towards some law, then we can simply remove it from the set of laws under consideration.

Assuming that the voter could cheat in the electoral process in order to obtain the parliamentary configuration they wishes to obtain, in this problem the goal will be deciding what would be the optimal electoral outcome for a given voter. Ideally, any voter should *easily* identify what would be the best result of an election according to their personal interests—so that they can focus on deciding what to vote to help that goal. However, in Section 3 it will be proven that just deciding the best electoral outcome, i.e. the best parliamentary seats distribution, is not easy at all in general.

In the decision version of problem, the minimum number of laws to get passed is part of the input of the problem, and the question is whether this number can be reached with some distribution of parliamentary seats.

Specification:

- Number of laws we want to pass: G.
- Number of seats in parliament: s.
- Parties: $\{P_1, \ldots, P_m\}$.
- Set of laws that the voter wants to get passed: $L = \{L_1, \ldots, L_n\}$. For each L_j with $j \in \{1, \ldots, n\}$:

[1] We may also consider that we want other laws to be *rejected*, but for the sake of notation simplicity we will assume that we want all of them passed. Note that the laws we want to get rejected could just be deemed in their negated form, so we want their *negation* to be *passed*.

[2] By *proxy decision-making we mean the fact that voters do not directly vote political decisions, but they vote for representatives who vote these political decisions according to their (previously announced) stances towards these decisions.*

[3] Note that this inversion trick does not yield a *fully* equivalent problem instance by itself, as the parliament must adopt some arbitrary tie-breaking policy (for instance, if yes and noes tie then the law does *not* pass), and this *asymmetric* treatment between yes and no matters in a perfectly tied voting. Still, the notation simplification achieved by assuming that all laws are desired by our voter justifies adopting that assumption.

  - A law is approved if and only if the number of votes in favour of it is strictly greater than the number of votes against it.
  - We define tuples F_i, $\forall i \in \{1,\dots,m\}$, as $F_i = (f_{i1},\dots,f_{in})$, where $\forall i,j$ with $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$, $f_{ij} \in \{-1,0,1\}$ means that party i is against, neutral (i.e. abstention), or in favour of law j, respectively.

- The solution space will be determined by the numbers of seats all parties get, restricted to the fact that all must add up to s. Each problem solution is a tuple $S = (s_1,\dots,s_m)$ where $\forall i$ with $i \in \{1,\dots,m\}$ we have that $s_i \in \{0,\dots,s\}$ is the number of seats party P_i has got, and $\sum_{i=1}^{m} s_i = s$.

Sometimes we will abuse the notation and define L_j also as the set of parties in favour of law L_j (i.e. $P_i \in L_j$ iff $f_{ij} = 1$).

Then, the goal of the decision version of problem PARLIAMENT is finding out whether there exists a solution S such that $\sum_{i=1}^{n} c_i \geq G$, where:

$$c_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{m}(f_{ij} * s_j) \geq 1 \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

$$\sum_{k=1}^{m} s_k = s \text{ and } \forall k \in \{0,\dots,m\}\ s_k \in \{0,\dots,e\}$$

In the optimization version of this problem, the aim is getting the maximum number of laws passed instead of just reaching a given number of them. Assuming the notation from the decision version, the goal of the optimization problem is formally defined as follows: instead of finding out whether there exists a solution S fulfilling $\sum_{i=1}^{n} c_i \geq G$, the solution S maximizing $\sum_{i=1}^{n} c_i$ is looked for. Let us remark that, in the simplest case, the best solution S could be to award all seats to a single party. However, in the usual case, no party's electoral program will perfectly match voter preferences. Therefore, more sophisticated seat allocations will be needed to maximize the number of laws that can be passed.

2.2. PRESIDENT problem

The political and voting characteristics of each country are different (e.g. [1]). In the previous problem, the aim was to obtain the parliament passing as many laws of those a given voter is interested in as possible. Next, the focus will move to elections where a single question is asked, electors can vote for several (not just binary) possible answers, and the goal is simply making some given answer win. For the sake of simplicity, let us consider that the question is which president should be elected, the possible answers are the candidates, and the goal is that a specific candidate gets elected.

Note that, in the PARLIAMENT problem presented previously, the goal was finding some power equilibrium indirectly yielding some outcome afterwards, as the numbers of parliamentary seats of parties affected which laws would be passed next. In the presidential election problem, called PRESIDENT, the goal is just making some candidate president, although this goal is also reached by some indirect means. Before the election, candidates can announce whether they will support some other candidate with the electoral power gained in the election if that power turns out to be insufficient to become president themselves. It will be assumed that candidates will only help candidates who received, in the election, more electoral power than themselves, and that this support is reciprocal, i.e. two or more like-minded candidates forming such alliance will support, after the election, the one of them getting more power in the election.

Votes turn into that *electoral power* as follows. It is supposed that the country is divided into a set of electoral constituencies, so that the most voted candidate in each constituency (state, province, district, etc.) gets *all* the representatives from that constituency. The candidate who gets the most votes from all the representatives nationwide (i.e. counting all constituencies) will be elected president. All the representatives gained by all the candidates in each of the candidate alliances explained before will actually vote for the allied candidate who received more representatives in the election.

Given the political setting described before, the PRESIDENT problem consists in deciding how a subset of voters supporting exactly the same candidate, and voting in different country constituencies, should vote to make their common candidate president (in the decision version) or to maximize the number of representatives voting for the candidate (in the optimization version). It is assumed that these voters perfectly know how the rest of the electorate will vote in all constituencies, that is, they know a perfect pre-election poll telling how all voters – but them – will vote in the election day.

How should these voters coordinate their votes to reach their common goal? If voting were a straightforward task indeed, then all of them should just vote for their common candidate, but this is not the optimal strategy in general. Note that, according to the pre-election polls, winning in some constituencies could be impossible for their common candidate, so voters in them should vote for other candidates being in alliance with their candidate, instead of for their candidate. Moreover, no candidate in that alliance should get, nationwide, more constituency representatives than their own candidate, because in that case their candidate will have to support another candidate. Hence, some voters should vote for candidates *out* of that alliance to reach their common goal. These difficulties will make the decision problem NP-complete, and the optimization problem inapproximable at *any* level, as it will be proven later.

In the formal definition of the problem, we will assume that there is a single alliance of candidates, and that it includes the candidate supported by the sub- set of voters to be coordinated. This particularization will not reduce generality to the hardness properties of PRESIDENTproven later: since NP-hardness and approximability hardness results trivially propagate by generalization, the hardness results given here will trivially apply to any problem version also allowing other alliances not involving the target candidate. Note, however, that generalizations can yield approximability hardness also in tougher approximability classes. This is not a possibility in this case, as it will be proved that the problem, as it is defined (i.e. with at most one alliance including the supported candidate), cannot be approximated to any extent. Also, note that the inclusion in class NP does not automatically propagate via generalization, so attention has to be paid to it. Fortunately, since finding out the winner of each alliance takes polynomial time (it just consist of a polynomial number of additions and comparisons), it is easy to see that the inclusion of the problem in NP, proved later, would not be affected by additionally allowing alliances not including the target candidate.

Let us adopt the point of view of the subset of voters coordinating to favour their common candidate. In the decision version of PRESIDENT, it has to be decided whether it is possible to coordinate our subset of voters in such a way that our candidate is elected president. It is assumed that being elected president requires receiving the votes of strictly more than a half of all constituency representatives nationwide.

Specification

- There are n states (i.e. constituencies).
- At each state $i \in \{1,\dots,n\}$ there are p_i voters of our subset of voters.
- There are m candidates.
- Our candidate is the j'th, and the set of candidates $M_{j'} \subseteq \{1,\dots,m\}$, with $j' \in M_{j'}$, form an *alliance*, i.e. each candidate $j \in M_{j'}$ will ask all constituency representatives won by them to support the candidate in $M_{j'}$ winning more constituency representatives.
- The number of representatives the winner of each state i gets is e_i.

- v_{ij} is the number of votes candidate j will have in state i, without taking into account the voters of our subset.
- In the event of a tie in the number of votes within a constituency, the representatives of the constituency will be won by the candidate whose name has the lowest lexicographical order. This untie criterion will also apply to decide which candidate must be supported by all candidates within an alliance, if several of them receive the same number of representatives. Since the lexicographical order is arbitrary, for the sake of notation simplicity it will be assumed that candidate j' is the one with the highest lexicographical order, and that all their allies are in a higher order than any of the remaining candidates. Thus, all ties will be resolved against our candidate j' (and next, against their allies).

Each problem solution will be defined by a set of values $x_{ij} \in \mathbb{N}$ for all $i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$ representing the number of voters in our voter subset who will vote for candidate j in state i.

The goal of the decision version of PRESIDENT is finding out whether some assignment of values to all variables x_{ij} satisfies the three constraints (3), (4), and (5) presented below. In the next expressions, the value of each term d_{ij} will be 1 if candidate j wins in state i and 0 otherwise, and $ord(j)$ returns the lexicographical order of candidate j. That is, the definition of d_{ij} is as follows:

$$d_{ij} = \begin{cases} 1 & \text{if } v_{ij} + x_{ij} \bigoplus_{jk} v_{ik} + x_{ik} \quad \forall k \in \{1, \dots, m\}, k \neq j, \text{ where} \\ & \bigoplus_{jk} = \begin{cases} \geq & \text{if } ord(j) < ord(k) \\ > & \text{otherwise} \end{cases} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Constraint (3) states that, in each state i, the total number of votes cast by the voters in our subset does not exceed the number p_i of voters of the subset in that state:

$$\sum_{j=1}^{m} x_{ij} \leq p_i, \forall i \in \{1, \dots, n\} \tag{3}$$

Constraint (4) requires that the number of constituency representatives *directly* won by our candidate j' (i.e. before considering the alliance) strictly beats the number of direct votes of any other candidate in the alliance. Since candidate j' is assumed to be the last one in lexicographical order, candidate j' is after all allied candidates in that order, and thus, in the event of a tie, all constituency representatives won by all candidates in the alliance will go to another candidate in the alliance. Hence, if there is a candidate k in the alliance getting at least as many representatives as candidate j', then candidate j' will be forced to support candidate k with all their representatives—and candidate j' will have no chance of winning the election:

$$\sum_{i=1}^{n} e_i d_{ij'} > \sum_{i=1}^{n} e_i d_{ij}, \forall j \in M_{j'} \tag{4}$$

Finally, constraint (5) indicates that candidate j' actually wins the election by getting strictly more than half of the total representatives, counting both the representatives directly won and those won by all their allies (which, by the second condition, will support candidate j'):

$$2 * \sum_{i=1}^{n} \left(\sum_{j \in M_{j'}} e_i d_{ij} \right) > \sum_{i=1}^{n} e_i \tag{5}$$

Note that, for each state $i \in \{1, \dots, n\}$, there will be at most a single $j \in M_{j'}$ such that $e_i d_{ij} > 0$, because each state has a single winner: if $e_i d_{ij} > 0$ then candidate j won that state. Also note that $j' \in M_{j'}$.

Let us remark that the winner of all the representatives of a state is the candidate who gets the most votes in the state, with no possibility of *local* alliances within the scope of a state. Alliances apply only at the nationwide level, and it is actually possible that the candidate who wins the most votes nationwide is not the one winning the most representatives, neither before nor after counting the additional representatives received by an alliance. For instance, let us suppose some candidate x gets *all* the votes in some large subset of constituencies providing overall less than half of the representatives and, in the remaining constituencies, x gets just one vote less than some other candidate x' winning them all. Then x' would be elected president, even though x could have more votes than x' nationwide.

Next we consider the optimization version of the problem. There are several reasons for aiming at maximizing the number of representatives—beyond the obviousness that they may let our candidate become president. On the one hand, if our candidate can win, then it is relevant to maximize the legitimacy of the government by getting as many representative votes as possible. On the other hand, if our candidate cannot, then it is interesting to show the strength of the eligibility in future elections.

The definition of the optimization version is as follows:

$$\begin{aligned} \max \quad & \text{if } \sum_{i=1}^{n} e_i d_{ij'} > \sum_{i=1}^{n} e_i d_{ij}, \forall j \in M_{j'} \\ & \text{then } \sum_{i=1}^{n} \left(\sum_{j \in M_{j'}} e_i d_{ij} \right) \\ & \text{otherwise } \ 0 \\ \text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} \leq p_i, \forall i \in \{1, \dots, n\} \end{aligned} \tag{6}$$

where d_{ij} is defined as in the decision version of the problem. The rest of the definition is also very similar. The main difference is that now it is not required that no ally gets more representatives than our candidate, but our candidate is just assigned 0 representatives in that case, because then they will have to support some other ally. Otherwise, our candidate will get the representatives of the states where they win as well as those of the states where their allies win, like in the decision version (recall that, for each state i, there is a single candidate j such that $d_{ij} > 0$).

3. Hardness of the problems

In this section it is proved that PARLIAMENT problem is NP-complete and cannot be approximated by any ratio $\rho' \leq \frac{1}{1-\frac{1}{e}}$. It is also proved that PRESIDENT is NP-complete and cannot be approximated by any ratio unless **P** = **NP**.

3.1. *PARLIAMENT problem*

Next, it will be proved the NP-completeness of this problem by making a polynomial reduction from the well-known NP-complete problem 3−SAT and next checking that the problem belongs to NP.

NP-Completeness of PARLIAMENT:

Given an instance of 3−SAT (a propositional formula in 3-CNF form), we want to produce an instance of PARLIAMENT such that the formula of 3−SAT is satisfiable if and only if we can make a seat distribution for the parties in PARLIAMENT such that at least G laws are passed.

Consider an instance of 3−SAT with n disjunctive clauses $F = \{F_1, \dots, F_n\}$ where $F_i = f_{i1} \vee f_{i2} \vee f_{i3}$, $\forall i \in \{1, \dots, n\}$, and each $f_{ij}, j \in \{1, 2, 3\}$, is a literal, i.e. it is either x or $\neg x$, where x is a Boolean variable. Suppose we have m variables, called $\{x_1, \dots, x_m\}$. Thus, each clause F_i is the disjunction of three of these variables in literal form (i.e. negated or not).

Now, we can define our instance of PARLIAMENT based on the instance of 3−SAT as follows.

The number of seats in the parliament equals the number of variables m used in the propositional formula, and the number of parties is $2m$. The party set is $P = \{p_1, \dots, p_m, \neg p_1, \dots, \neg p_m\}$. The set of laws L

consists of $n+m$ laws, and is defined as $L = \{L_1, \dots, L_n, C_1, \dots, C_m\}$, where each set L_i (as we said before, this is the set of parties in favour of law L_i in a notation abuse) is given by $L_i = \{l_{i1}, l_{i2}, l_{i3}\}$, where

- $l_{ij} = p_k$ if $f_{ij} = x_k$ for some $k \in \{1, \dots, m\}$; or
- $l_{ij} = \neg p_k$ if $f_{ij} = \neg x_k$ for some $k \in \{1, \dots, m\}$.

The sets C_k, $k \in \{1, \dots, m\}$, are defined as $C_k = \{p_k, \neg p_k\}$ $\forall k \in \{1, \dots, m\}$ (that is, the parties in favour of law C_k are p_k and $\neg p_k$). The number of laws to pass, i.e. G, is $n+m$, so we need to pass all laws in this problem instance.

We still have to set the stance of parties towards laws they are not in favour of according to the previous requirements. If a party is not in favour of some law, then it will abstain when that law is voted. That is, no party will be against any law in this problem instance.

Now, we will prove that the proposition formula of 3−SAT is satisfiable if and only if we can pass all the laws in our instance of PARLIAMENT.

Proof of (if) part

Given a truth assignment T that maps every variable to *True* or *False* and makes all the clauses of F true, we want to prove that there is a distribution of seats among parties such that at least $n+m$ laws are passed (i.e. all laws are passed).

Let us suppose function $wLaw : L \to \mathbb{Z}$ returns, for each law, the difference between the number of seats that are in favour and against a given law, that is:

$$wLaw(i) = \sum_{j=1}^{m} (f_{ij} * s_j) \tag{7}$$

Let us consider a distribution of seats where each party p_k, $k \in \{1, \dots, m\}$, is given one seat if $T(x_k) = True$, and each party $\neg p_k$, $k \in \{1, \dots, m\}$, is given one seat if $T(x_k) = False$. Note that m seats are delivered, as required. Let us prove that all the laws in our PARLIAMENT instance are passed with this distribution of seats.

Since truth assignment T satisfies all disjunctive clauses F_i, $i \in \{1, \dots, n\}$, for each clause F_i there exists $j \in \{1, 2, 3\}$ such that $f_{ij} = x_k$ for some $k \in \{1, \dots, m\}$ and $T(x_k) = True$, or such that $f_{ij} = \neg x_k$ for some $k \in \{1, \dots, m\}$ and $T(x_k) = False$. In the former case, party p_k supports law L_i and has one seat, and in the latter, party $\neg p_k$ supports law L_i and has one seat. No party is against any law, so $wLaw(L_i) \geq 1$. This applies to all laws L_i, so all these n laws are approved.

Now, let us see that laws C_k, $k \in \{1, \dots, m\}$, are approved. Since T is a truth assignment, for each Boolean variable x_k it follows that $T(x_k) = True$ or $T(x_k) = False$. In the former case, party p_k has one seat, and in the latter case, party $\neg p_k$ has one seat. We conclude $wLaw(C_k) = 1$, so law C_k is approved. This happens for all laws C_k, $k \in \{1, \dots, m\}$, so all of them are approved.

Then, all the $n+m$ laws in our PARLIAMENT instance are approved.

Proof of (only if) part

We assume that all the $n+m$ laws of our instance are passed. This implies that the m laws of the form C_k, $k \in \{1, \dots, m\}$, are approved. The m seats are distributed among parties, and the only way to get all of these laws passed is by delivering exactly one seat to each pair of parties p_k and $\neg p_k$, i.e. one of them gets one seat and the other none. From this distribution we can construct a truth assignment T such that, if p_k has a seat, then $T(x_k) = True$, and if $\neg p_k$ has a seat, then $T(x_k) = False$.

In addition, our seat distribution gets all laws L_i approved. By reasoning similarly as in the other implication, this means that all clauses F_i are satisfied with truth assignment T, and it can be concluded that our truth assignment T satisfies the propositional formula $F_1 \vee \dots \vee F_n$.

We need to see that the reduction is actually performed in polynomial time, which is evident since each element created for our PARLIAMENT instance has the same size or double size as some element in the original 3−SAT instance, and constructing the former from the latter is trivial.

Therefore, since 3−SAT is an NP-hard problem, we obtain that the PARLIAMENT problem is NP-hard.

We must also check that PARLIAMENT is in NP, but this is straightforward, since given a candidate solution, only a polynomial number of simple arithmetic operations of addition, multiplication, and comparison is necessary to know whether this solution is correct and reaches the minimum number of laws to be passed.

With this we can conclude that PARLIAMENT is an NP-complete problem. Let us now analyse the approximability of the problem.

Given an optimization problem, the *performance ratio* of a given solution for a given problem instance is $max\left\{\frac{sol}{opt}, \frac{opt}{sol}\right\}$, where sol is the value of that solution for that instance and opt is the value of the optimal solution for that instance. By picking the maximum of both fractions, we make sure all performance ratios are within the same range $[1, \infty)$ regardless of whether the problem under consideration consists in a maximization or a minimization. This way the approximability of different problems is easier to compare regardless of their type (note that, given the maximization nature of our optimization problem, in our case the performance ratio will be $\frac{opt}{sol}$). Given an optimization problem and $r \in \mathbb{R}$, we say that a polynomial-time approximation algorithm provides an r-approximation to the problem if its worst (i.e. biggest) performance ratio for any instance is lower than or equal to r.

Next we prove an inapproximability threshold of this optimization problem by using a known inapproximability result of the Maximum Set Coverage problem.

Approximability of PARLIAMENT:

The *Maximum Set Coverage* problem consists in obtaining the largest number of elements of a set by picking a given number of predefined subsets of this set.

The problem instance consists of the following elements:

- S = $\{S_1, \dots, S_m\}$ is the set of available subsets, where $\bigcup S$ is the set of all the elements.
- $k \in \mathbb{N}$ is the number of subsets we can pick.

This optimization problem cannot be polynomial-time approximated with a ratio better than $\rho = \frac{1}{1-\frac{1}{e}}$ unless **P** = **NP** (see [14]), where e is Euler's number.[4] Thus we cannot expect any polynomial-time algorithm to guarantee solutions whose value is always better than $\simeq 63\%$ of the optimal value.

A strict polynomial-time reduction f is presented which, given any *Maximum Set Coverage* instance I, constructs an instance of the PARLIAMENT problem $f(I)$ as follows:

- k is the number of seats of the parliament.
- The elements in $\bigcup S$ are the laws we want to approve.
- Each subset S_i, $i \in \{1, \dots, m\}$, is a party, and the elements of S_i are the laws that party is in favour of.
- No law has any party against it.

Terms opt and opt' will be used to refer to the optimal solutions of *Maximum Set Coverage* instance I and PARLIAMENT instance $f(I)$, respectively.

The reduction will prove that PARLIAMENT is impossible to approximate for any ratio $\rho \leq \frac{1}{1-\frac{1}{e}}$ unless **P** = **NP**. The reason is that *Maximum Set Coverage* has the same inapproximability threshold, and the polynomial reduction we are constructing between both problems is *strict*. This means that, by using that reduction, the performance ratio for the original problem instance (in our case I) will always be *better*

[4] In [14] that ratio is given as $1-\frac{1}{e}$ because the performance ratio is defined as $\frac{sol}{opt}$ there.

than or equal to the performance ratio for the instance of the target problem (in this case $f(I)$). Hence, achieving a better performance ratio in the latter problem would be contradictory.

Let T' be any solution for instance $f(I)$ of the PARLIAMENT. From this solution, let us construct a solution T for instance I of *Maximum Set Coverage* as follows. Let $T' = (t'_1, \dots, t'_m)$ be the numbers of seats of parties. That is t'_i, $i \in \{1, \dots, m\}$, is the number of seats of party S_i. Then, we define *Maximum Set Coverage* solution T as the set of all subsets S_i such that $t'_i > 0$. If less than k subsets can be picked by using this criterion, then we fill T up to gathering k subsets by taking additional subsets in any arbitrary order (e.g. lexicographical).

It is clear that the *optimal* solutions of both problem instances consist in picking exactly the same collection of subsets or parties with at least one seat, respectively. Since no party is against any law, giving a single seat to a party guarantees that all laws it supports will be passed, so there is no necessity to use more seats on a party. Note that if we give only one seat to each party then we have equivalent problems, so that $opt = opt'$.

Let TI be the set of subindexes of the parties that had at least one seat in PARLIAMENT solution T', i.e. $TI = \{i \mid t'_i > 0\}$.

Let ρ be the performance ratio of solution T for *Maximum Set Coverage* instance I, and ρ' be the performance ratio of solution T' for PARLIAMENT instance $f(I)$. Then,

- $\rho = \dfrac{opt}{|(\bigcup_{i \in TI} S_i) \cup (\bigcup_{S_i \in T, i \notin TI} S_i)|}$, and
- $\rho' = \dfrac{opt}{|\bigcup_{i \in TI} S_i|}$

Clearly $\rho \leq \rho'$, so we have a strict reduction from *Maximum Set Coverage* to PARLIAMENT. This means that any inapproximability result of the former problem directly maps to the latter, so we can conclude that the problem PARLIAMENT is inapproximable for any ratio $\rho' \leq \frac{1}{1-\frac{1}{e}}$ in polynomial time unless **P** = **NP**. Note that this also implies that this problem cannot belong to approximation class **PTAS** if **P** ≠ **NP**.

3.2. PRESIDENT problem

In order to prove the NP-hardness of the problem, a well-known NP-complete problem will be polynomally reduced into PRESIDENT. The positive version of SUBSET SUM consists in, given a multiset S of integers greater than zero and an integer T greater than zero, finding out if there exists a subset S' of S such that the sum of the integers of S' is exactly S. Note that we can assume $T \geq s \ \forall s \in S$, since no s being strictly greater than T will ever be part of a solution. Recall that all numbers are positive, so removing from S all $s \in S$ being greater than T yields an instance having exactly the same solutions as before.

NP-Completeness of PRESIDENT:

Given an instance of SUBSET SUM consisting of:

- $S = \{s_1, \dots, s_l\}$, where $\forall i \in \{1, \dots, l\}\ s_i \in \mathbb{N}$.
- $T \in \mathbb{N}$ with $T \geq s_i$, $\forall i \in \{1, \dots, l\}$.

A PRESIDENT instance is constructed with:

- $n = l + 3$ states.
- $m = 3$ candidates $\{1, 2, 3\}$ respectively called A, B, and C.
- $j' = C$ and $M_{j'} = \{B, C\}$, so C is our candidate and B is our ally. The other candidate, A, will be called our *rival*.
- $\forall i \in \{1, \dots, l\}$, $e_i = s_i$. Let $\bar{S} = \sum_{i=1}^{l} s_i$. We have $e_{l+1} = \bar{S} + T$, $e_{l+2} = \bar{S} - T$, and $e_{l+3} = \bar{S} + 1$.
- $\forall i \in \{1, \dots, l\}$, $v_{iA} = 2$, $v_{iB} = 2$, and $v_{iC} = 0$. Besides,
  - $v_{l+1\,A} = 1$, $v_{l+1\,B} = 0$, $v_{l+1\,C} = 0$,
  - $v_{l+2\,A} = 0$, $v_{l+2\,B} = 1$, $v_{l+2\,C} = 0$,
  - $v_{l+3\,A} = 0$, $v_{l+3\,B} = 0$, $v_{l+3\,C} = 1$.
- $\forall i \in \{1, \dots, l\}$, $p_i = 1$ and $p_{l+1} = p_{l+2} = p_{l+3} = 0$.

Let us explain these numbers. Since a polynomial reduction from SUBSET SUM is performed, our objective is that there exists a solution for our instance of SUBSET SUM iff there exists a solution for the previous instance of PRESIDENT. The candidates actually winning states $1, \dots, l$ will denote the natural numbers taken to form a solution in the SUBSET SUM problem, let us see how. Note that there is a single voter of our subset of voters in each of these states i. With a single vote, this voter can tip the balance and give all the s_i representatives of the state to our ally B (if voting for B) or to our rival A (if voting for A or abstaining). These two choices will represent that number s_i is taken for the solution of SUBSET SUM or not, respectively.

On the contrary, the winners in states $l+1$, $l+2$, and $l+3$ are decided beforehand, because the number of voters of our subset in these states is 0 and the other voters will make A, B, and C win in them, respectively. Hence, before deciding the winner of the states 1 to l, we know that our rival A will have $\bar{S} + T$ representatives, our ally B will have $\bar{S} - T$, and our candidate C will have $\bar{S} + 1$. Therefore, the total number of representatives in this instance of PRESIDENT is

$$\sum_{i=1}^{l} e_i + e_{l+1} + e_{l+2} + e_{l+3} = \bar{S} + \bar{S} + T + \bar{S} - T + \bar{S} + 1 = 4\bar{S} + 1 \tag{8}$$

Hence, to win the election we need $2\bar{S} + 1$ representatives.

Making C president implies not tying with either B or A, because then the lexicographical order will be against C. In the problem instance we have constructed, our subset of voters cannot make C win any additional representatives beyond the $\bar{S} + 1$ representatives C will get from state $l + 3$ anyway. Thus, the only chance to make C win consists in delivering states 1 to l between A and B in some clever way. In fact, in order to become president, C will need that B gets *exactly* $\bar{S}$ representatives. If B gets more than that, then B will at least tie with C, so C will have to support B. On the other hand, if B gets less than that, then B and C combined will reach at most $2\bar{S}$ representatives, so the remaining at least $2\bar{S} + 1$ representatives will be for A, and A will be elected president. In order to make B get exactly $\bar{S}$ representatives, B will need to get exactly T representatives from the first l states. Since the representatives of each state are won on a all-or-nothing basis, winning exactly T representatives in these states implies finding a combination of numbers among $s_1, \dots, s_l$ adding exactly T, which solves the SUBSET SUM instance.

With this in mind, we are going to prove that there is a solution for the SUBSET SUM instance if and only if there is a solution for our PRESIDENT instance.

Proof of (if) part

Suppose that we have a solution for our instance of SUBSET SUM i.e., we have a subset $S' \subseteq S$ such that $\sum_{s \in S'} s = T$. Let $WS = \{i \mid s_i \in S'\}$. Since $\forall i \in \{1, \dots, l\}$ we have $e_i = s_i$, we infer $\sum_{i \in WS} e_i = T$. Hence, if we use the voters in our subset of voters to make B win exactly the states in set WS and let A win all the others, then B will get exactly $(\bar{S} - T) + T = \bar{S}$ representatives. As stated above, in this case C will be elected president, so this is a solution for our instance of PRESIDENT.

Proof of (only if) part

Suppose that we have a solution for our instance of PRESIDENT. Let $WS = \{i \mid 1 \leq i \leq l \wedge B \text{ wins state } i\}$ and consider $S' = \{s_i \mid i \in WS\}$. By following the same reasoning as before we infer $\sum_{s \in S'} s = T$, so S' is a solution for our instance of SUBSET SUM.

It is clear that this reduction can be done in polynomial time. In particular, the amounts of voters, candidates, and states of the resulting PRESIDENT instance are all polynomial with the size of the original SUBSET SUM instance. Since SUBSET SUM is NP-hard [15], we conclude that PRESIDENT is also NP-Hard.

Besides, PRESIDENT belongs to NP, because we can find out whether a candidate solution to the problem makes j' win or not

in polynomial time (only a polynomial number of comparisons, additions, and multiplications are needed). Therefore, PRESIDENT is NP-complete.

Let us analyse if it is possible to find good approximate solutions in a reasonable amount of time, in particular polynomial time. As it will be proven next, it is very unlikely.

Let us prove that for any function $r : \mathbb{N} \to \mathbb{R}^+$, if a polynomial-time algorithm can approximate the PRESIDENT problem with a performance ratio of $r(x)$, where x is the size of the problem instance, then **P** = **NP**.

Before starting the construction, the NP-complete problem PARTITION is introduced (see [16]). This problem consists in, given a multiset of positive integers S, deciding whether we can find two subsets S_1 and S_2 such that $S_1 \bigcup S_2 = S$, $S_1 \bigcap S_2 = \emptyset$, and $\sum S_1 = \sum S_2$.

Note that a polynomial-time $r(x)$-approximation algorithm for PRESIDENT guarantees a solution whose value is better than 0 for any instance whose optimal solution has a value higher than 0, because $r(x) \in \mathbb{R}^+$ is an upper bound of the worst-case (i.e. highest) value of $\frac{opt}{sol}$ for any instance. As we will see, we will reduce the decision PARTITION problem to the maximization PRESIDENT problem in such a way that, if the optimal solution for the resulting PRESIDENT instance has a value different from 0, then any solution giving a value different from 0 will be optimal. Hence, the solutions returned by any $r(x)$-approximation algorithm for PRESIDENT will be optimal for these PRESIDENT instances. We will also see that the optimal solutions for these particular PRESIDENT instances will provide correct answers to the original instances of the NP-complete PARTITION problem. Thus, by applying the reduction and next the $r(x)$-approximation algorithm for PRESIDENT we could solve any PARTITION instance in polynomial time, which would imply **P** = **NP**.

Inapproximability of PRESIDENT:

Let us consider an instance of PARTITION $S = \{s_1, \ldots, s_n\}$ with $s_i \geq 0, \forall i \in \{1, \ldots, n\}$. Without loss of generality we assume that $\sum_{i=1}^{n} s_i$ is even (otherwise the answer to PARTITION is trivial). We construct the PRESIDENT instance as follows:

- There are three candidates $\{1, 2, 3\}$ with names A, B, and C, where $j' = C$ is our candidate and $M_{j'} = \{A, B, C\}$, that is, all candidates are allies.
- There are $n + 1$ states.
- $e_i = s_i \ \forall i \in \{1, \ldots, n\}$, and $e_{n+1} = T + 1$ where $T = \frac{\sum_{i=1}^{n} s_i}{2}$.
- $v_{iA} = v_{iB} = 2 \ \forall i \in \{1, \ldots, n\}$, and $v_{n+1\,A} = v_{n+1\,B} = 0$.
- $v_{iC} = 0 \ \forall i \in \{1, \ldots, n\}$, and $v_{n+1\,C} = 1$.
- $p_i = 1 \ \forall i \in \{1, \ldots, n\}$, and $p_{n+1} = 0$.

Note that, before deciding how our subset of voters will vote, our candidate C knows that they will directly win the $T + 1 = \frac{\sum_{i=1}^{n} s_i}{2}$ representatives of state $n+1$ in any case, and directly winning any other state will be impossible. On the other hand, candidates A and B do not have any representative guaranteed beforehand. The remaining n states have altogether $2T$ representatives to be delivered between A and B depending on the choices of our subset of voters. If either A or B reach at least $T + 1$ representatives from some of these states, then candidate C will have to ask their representatives to support that ally, and thus C will get the votes of 0 representatives. Thus, candidate C will only get more than 0 representative votes if the $2T$ representatives of states 1 to n are evenly delivered between A and B, each one receiving T representatives. In this case, A and B will support C, and C will win the votes of *all* the $3T + 1$ representatives. As we see, C can get either all representative votes or 0 representative votes, depending on whether the original PARTITION numbers $s_1, \ldots, s_n$ can be evenly split in two subsets or not, respectively.

Now, suppose that there exists an algorithm $\mathcal{A}$ that can solve PRESIDENT in polynomial time with a ratio $r(x) \in \mathbb{R}^+$. Let us suppose that we run $\mathcal{A}$ for the PRESIDENT instance derived from the original PARTITION instance as explained. There are two possibilities:

(a) The algorithm returns a solution with 0 value. Let us see that this solution is optimal.
Let us suppose that it is not, i.e. the optimal solution for this instance of PRESIDENT actually has value $s > 0$. Since $r(x) \geq \frac{opt}{sol}$ in all cases and $opt = s$, the approximation algorithm $\mathcal{A}$ must return a solution sol whose value is higher than or equal to $\frac{s}{r(x)}$. For any $r(x) \in \mathbb{R}^+$, this implies that the algorithm cannot return any 0-value solution. Therefore, we get a contradiction.
We infer that the actual optimal solution for this PRESIDENT instance has 0 value, so the solution returned by the algorithm is optimal. Therefore, there does not exist a distribution of states between A and B that gives each one T representatives (otherwise, with this distribution C would get more direct representatives than each of them and both would have to support C, thus letting C achieve $3T + 1 > 0$ representative votes at the end). By the construction of the PRESIDENT instance from the PARTITION instance, we conclude that S cannot be divided into two subsets that add up to the same value, so the answer to the original PARTITION instance is 'no'.

(b) The algorithm returns a solution whose value is higher than 0.
Note that there is no solution for the PRESIDENT instance under consideration whose value is higher than 0 and lower than $s = 3T + 1$: the only solution giving more than 0 value is obtained when A and B tie at T representatives each (otherwise one of them would at least tie with C, and C would have to support A or B by the untie convention, thus getting 0 representative votes). Hence, the solution returned by algorithm $\mathcal{A}$ must have value $sol = 3T + 1$. In this solution, let terms $x_{ij} \in \mathbb{N}$ denote the number of voters in our subset of voters who vote for candidate j in state i. Then, a positive solution for the original PARTITION can be constructed by choosing S_1 and S_2 as follows: $S_1 = \{s_i \mid x_{iB} = 1\}$ and $S_2 = \{s_i \mid x_{iC} = 1\}$. Since candidate C cannot be supported by A and B unless both of them tie, we infer $\sum_{v \in S_1} v = T = \sum_{v \in S_2} v$, so this is a solution for original PARTITION instance, and the answer to this instance is 'yes.'

We conclude that, if algorithm $\mathcal{A}$ exists, then PARTITION can be decided in polynomial time: we transform the PARTITION instance into a PRESIDENT instance as defined by the polynomial reduction, next we run $\mathcal{A}$, and finally we return 'yes' iff the solution returned by $\mathcal{A}$ for the PRESIDENT instance has more than 0 value. Since PARTITION is NP-complete, this polynomial-time solution of PARTITION would imply **P** = **NP**. That is, if **P** ≠ **NP** then $\mathcal{A}$ cannot exist.

4. Experimental results

Once it has been proven that the problems are inapproximable up to some level in the general case, a genetic algorithm will be provided to obtain reasonable solutions in reasonable execution times. Notice that, according to the considerations given in [17], the approximation hardness of a given NP-hard problem should affect the design decision of what algorithm is used to approximately solve it. For instance, depending on that approximability, it would be suitable to use a specific-purpose greedy algorithm with some performance ratio guarantee, a metaheuristic without that guarantee but performing well on average, etc. Accordingly, we choose Genetic Algorithms to find reasonable sub-optimal solutions for both problems under consideration. Indeed, stochastic local search metaheuristics (e.g. [18–21]), and Genetic Algorithms in particular, have proven their usefulness to deal with many problems with bad approximability (see e.g. [22–26]).

A genetic algorithm is a solution search method often applied to optimization problems (see [27,28]). These algorithms emulate the evolution of species through the survival of the fittest. A population of problem solutions codified in some way (e.g. vectors of bits or natural numbers) evolves through the algorithm execution. A *fitness function*, determining how good each solution is, is used to select

the fittest of them and discard the rest, which are replaced by new solutions. In order to make new solutions similar to previous ones, each new solution is created by mixing two existing solutions similarly as crossover works in nature. In addition each piece (gene) of each solution has some probability of receiving a new random, non-inherited value. Note that this mutation step is performed at each iteration of the genetic algorithm, with the aim of being able to extend the exploration of the search space.

4.1. *PARLIAMENT problem*

The algorithm works as follows:

1. Each solution in the initial population is created as follows. Initially, 0 seats are assigned to all parties, next a random party is chosen and given 30 seats, and this process is repeated until there are no seats left (if there are less than 30 seats remaining, all the remaining seats are assigned to the chosen party).
2. Each solution (chromosome) will denote the distribution of seats of all parties, provided that the sum of the seats for each party is equal to the total number of seats. Each gene is a natural number denoting the number of seats assigned to some party. When new solutions are created from two other solutions, for each party the number of seats in the new solution is the average of the number of seats in the parent solutions rounded down to the nearest integer. Since the seats of our solution must add up to 350 in our running example, and some seats will be lost by making these integer divisions, some seats remain after this procedure. The remaining seats are assigned randomly. For each new solution to be created in subsequent algorithm steps, initially two solutions are randomly selected from the population. The probability of picking each solution is proportional to its fitness (i.e. the roulette selection method is adopted). Then, the new solution is created by crossover as stated above. Each gene of the new solution can mutate according to 4 different mutation methods (explained below). After some previous experiments, the probability of each gene to mutate was set to 10% as it was observed to provide the best results.
3. When the maximum number of iterations is reached, the algorithm stops and returns the solution with the highest fitness value.

Four different types of mutation are used:

- **mut1**: Selecting two parties from the solution and exchanging the number of seats they hold.
- **mut2**: Selecting a party which will absorb all the seats of another party.
- **mut3**: Selecting a party that will absorb at most 17 seats from 4 randomly selected parties.[5]
- **mut4**: Selecting one of the three mutations mentioned above at random with the same probability.

In order to assess the usefulness of the approach in the search for solutions of the PARLIAMENT optimization problem, a real case study will be faced. In particular, the Spanish parliament and the laws that were debated in it during the parliamentary term between 2019 and 2023 will be considered, taking into account the actual stances defended by the political parties of said parliament for each law. The data have been obtained from [29].

Several problem instances have been created, all of them concerning the same 27 propositions of Law and Bills voted in the Congress of Deputies of Spain, a parliament with 350 seats where the main 8 political parties were PSOE, PP, VOX, UP, ERC, Cs, PNV, and EH Bildu in the term stated before. These bills were made by different political parties. Since all votings are registered, for each proposition the stance held by each of the parties towards each proposition is known.

From this common setting, different problem instances are considered, each one differing in the subset of propositions wanted to be approved by the parliament. For each instance, the corresponding subset includes propositions of different parties, so that there are not trivial solutions where a single party could defend and approve all the required laws on its own. In fact, we also include an instance where all 27 laws are wanted to be approved.

Since 350 seats have to be distributed among 8 different political parties, exhaustively analysing all the $\binom{350+8-1}{8-1}$ possible combinations is unfeasible. Let us remark that this combinatorial number is approximately $1.38e14$, so the size of problem instances justifies the use of genetic algorithms to heuristically solve them.

To collect and evaluate experimental results, each genetic algorithm is run 1000 times, and in each run, 500 iterations and an initial population of 20 different seat arrangements.[6] Table 1 and Fig. 1 summarize the results obtained for each problem instance, comparing the usefulness of the four different mutations. As it can be seen, **mut3** wins in almost all the instances. This is mainly due to the fact that voting groups are usually formed for some large subsets of laws, and in particular it happens a lot that there are two antagonistic groups of parties voting the opposite for many laws. There are always one or two parties that may differ, but not much more. Genetic algorithms that focus on exploiting good solutions (as opposed to exploring more

Table 1
Results obtained in the PARLIAMENT problem in four instances with different sets of laws to be passed. For each mutation type it is shown the minimum, maximum and average number of laws passed. Mutations **mut3** and **mut4** obtain the best results.

Instance 1: 7 laws				
Mutation	Min	Max	Mean	Variance
mut1	4	4	4	0
mut2	4	6	4.03	0.04
mut3	6	7	6.99	0.01
mut4	7	7	7	0

Instance 2: 11 laws				
Mutation	Min	Max	Mean	Variance
mut1	6	8	7.12	0.20
mut2	7	9	7.80	0.33
mut3	8	10	9.66	0.23
mut4	9	10	9.01	0.01

Instance 3: 18 laws				
Mutation	Min	Max	Mean	Variance
mut1	13	14	13.64	0.23
mut2	14	15	14.63	0.23
mut3	15	16	15.44	0.25
mut4	14	15	14.99	0.01

Instance 4: 27 laws				
Mutation	Min	Max	Mean	Variance
mut1	19	21	20.41	0.69
mut2	21	23	21.96	0.26
mut3	21	25	22.95	0.17
mut4	22	24	23.11	0.18

[5] This mutation enables a wide exploration of the solution space. The idea is that a party sometimes takes 5% of the total seats in the parliament from other parties.

[6] Preliminary tests with fewer iterations indicated that the results could be very poor, while higher numbers did not significantly improve the results for the extra time needed for execution.

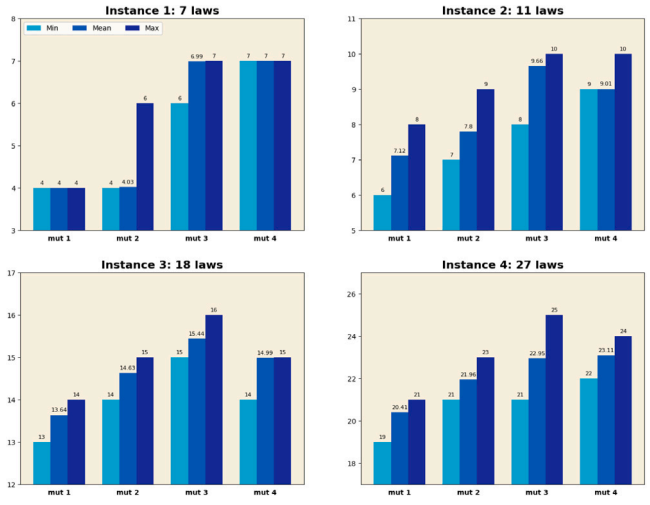


Fig. 1. Results obtained in the PARLIAMENT problem. Graphical view of the data shown in Table 1.

diverse solutions) find it difficult to find better solutions from some iteration on, because some sub-optimal solutions easily found with little effort are difficult to improve with local changes. For example, if we gave almost all the seats to one of the two typical antagonistic groups, then we could get a decent number of laws passed, but getting out of these solutions is complicated, as improving them requires several iterations of (locally worsening) seats modifications. On the other hand, by using the third type of mutation (which involves more parties in a single modification), the exploration of very different solutions is greatly encouraged. Actually, as can be seen, by using **mut3** better solutions are reached for instance 1, where all 7 laws are passed in all algorithm executions despite the fact very poor results are reached with **mut1** and **mut2**. It is also interesting to mention that **mut4** beats **mut3** in the last instance (the one where getting all laws and bills is wanted, i.e. the hardest one). Being able to use *also* **mut1** and **mut2** enlarges the repertory of available seats modifications, which in turn increases the set of possible sequences of consecutive modifications where all steps are locally beneficial. Note that some of these additional sequences could yield some good *complex* seats modifications that could be very difficult to form otherwise. We think this increased tendency to exploration is critical to successfully handle instance 4.

We conclude that we can find decent solutions to the PARLIAMENT problem in an efficient way by using genetic algorithms performing some exploration of the solution space.

4.2. *PRESIDENT problem*

In order to study the practical resolution of the PRESIDENT optimization problem, sub-optimal solutions are found by using a genetic algorithm. Initially it might seem unproductive to apply genetic algorithms to this problem, given the large size of the solution space and the big number of *null* solutions, that is, those where our candidate gets zero representatives because an ally gets more representatives. However, we can proceed as follows:

- If our candidate j' can win some state by being voted by all our voters in that state, then we directly assume j' wins. Note that a victory of j' in the state is always at least as good as a victory of any other candidate, no matter if it is ally or not: in either case, giving the state to j' can only help to make j' the most voted candidate in their alliance, as well as to make j' beat any non-ally candidate. Since voters in a state are useful only in that state, there is no reason for strategic voting in this case, and the straightforward choice of voting j' in that state is optimal.
- Although we can send our voters in some state to vote for several different candidates, only the winner in the state matters. Hence, instead of choosing one by one how each of our voters in that state must vote, we can just choose the candidate our voters will make win—from the subset of candidates who can actually win the state with our voters. Therefore, the solution space to be explored by the genetic algorithm is the set of all tuples where the first component is any candidate who can win the first state with our votes, the second component is any feasible winner of the second state, and so on for all states.[7]

Hence, each solution in the genetic algorithm is just a tuple defining the winners at all states. States where our candidate can win do not need to be represented by any component in the tuple and can be fully ignored by the algorithm, as they can be directly assigned to our candidate in any solution. The crossover operation of the genetic algorithm works as follows: given two solutions, the winner of each state in the child solution is defined by picking the winner of the same state from one of the two solutions picked at random. The mutation operation changes the winner of a state by any other possible winner at that state. The algorithm works as follows:

1. In the initialization, solutions are constructed where our candidate j' is made winner in all states where it is possible. These genes cannot be improved, so they are fixed to improve the behaviour of the algorithm. However, in each solution in the initial population, for all states where our candidate j' cannot win, any of the possible state winners is randomly assigned.
2. In each iteration, pairs of solutions are picked at random in such a way that the solutions are distributed proportionally with respect to the fitness function, so better solutions are more likely to get selected, and new solutions are generated from each pair by crossing them as stated above.
3. In each of these new solutions, the winner of each state has a probability to mutate as stated above.
4. If the iteration limit is reached then the fittest solution is returned, else we get back to step 2.

Problem instances were randomly generated for different combinations of numbers of candidates, numbers of allied candidates supporting our candidate j', and numbers of states. For each instance, the number of a priori votes received by candidate j' (i.e. from voters *not* in our subset of voters) randomly varies between 10 and 25 in each state, and for the rest of candidates, this number is randomly chosen between 10 and 30 in each state. Finally, the number of voters whose vote can be chosen in the solutions to help their common preferred candidate is chosen at random between 0 and 10 for each state. Also, the number of candidates, allies and states were decided beforehand (i.e. not randomly), as they represent the size of the problem under consideration.

In all cases, the algorithm was run with a population of 20 solutions and for 500 iterations. Experiments showed this number lets achieve way better results than smaller ones, though bigger ones do not make solutions significantly better. For each instance, the algorithm is run 1000 times, and the maximum value, the minimum (without counting null solutions), the number of null solutions, the mean, and the variance are collected. In addition, we compare the results with the total number of representatives in all states.

[7] Moreover, if we only want to maximize the number of representatives that will support our candidate regardless of the support received by the remaining candidates (which is indeed the definition of the optimization version of PRESIDENT), then all the candidates not being in alliance with our candidate can be deemed as a single candidate, because the distribution of representatives among them is irrelevant for that goal.

Table 2
Results of the PRESIDENT problem considering four instances with different numbers of candidates, allies, and states. For each of them, it can be seen the result obtained with different mutation ratios after repeating experiments 1000 times for each configuration of parameters. Lower mutation ratios obtain better results.

Instance 1: 588 total repres.; 20 candidates; 7 allies; 40 states					
Mut%	Min	Max	#Null	Mean	Variance
1	531	531	0	531	0
2	531	531	0	531	0
5	531	531	0	531	0
10	521	531	0	530.83	1.67
15	504	531	0	524.46	39.34

Instance 2: 649 total repres.; 30 candidates; 5 allies; 45 states					
Mut%	Min	Max	#Null	Mean	Variance
1	487	527	0	521.73	65.63
2	499	527	0	525.73	18.52
5	501	527	0	524.25	26.55
10	445	517	0	482.72	137.94
15	410	488	0	445.08	163.40

Instance 3: 1045 total repres.; 25 candidates; 10 allies; 70 states					
Mut%	Min	Max	#Null	Mean	Variance
1	908	908	0	908	0
2	908	908	0	908	0
5	908	908	0	908	0
10	853	908	0	886.37	100.35
15	811	898	0	851.06	171.93

Instance 4: 1514 total repres.; 20 candidates; 5 allies; 100 states					
Mut%	Min	Max	#Null	Mean	Variance
1	1220	1250	0	1247.10	31.33
2	1211	1250	0	1242.00	61.02
5	1125	1226	0	1175.31	244.81
10	1028	1143	0	1083.56	357.65
15	974	1097	0	1023.21	421.42

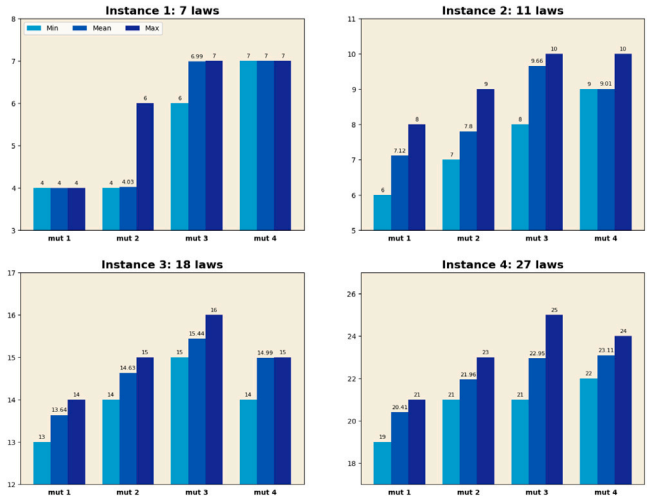


Fig. 2. Results obtained in the PRESIDENT problem. Graphical view of the data shown in Table 2.

Table 2 and Fig. 2 summarize the results for different instances with these mutation percentages: 1%, 2% 5%, 10%, and 15%. From these experiments it can be seen that the lower the percentage of mutation, the better the results. This is due to the fact that finding good solutions for this problem is easier by relying on the exploitation of good solutions, instead of on the exploration as it was done in the PARLIAMENT problem. In the last three instances, it can be observed that the lowest value observed for the 5% mutation rate is better than the highest value for 10% and 15%. In fact, for all instances and all mutation rates, solutions are found that are, at least, close to 2/3 of the total number of representatives, despite the fact that there may be many candidates, and that our candidate has a lot of difficulty to win in any state.

Note that these instances are large enough to make the use of branch-and-pruning algorithms to find the optimal solution unfeasible. In fact, there are $|M_{j'}|^n$ different solutions in the search space, being n the number of constituencies and $|M_{j'}|$ the size of the alliance of the candidate. According to the experimental data, the proposed algorithm generally returns very good solutions that are close to getting all the representatives in the states, and never obtains null solutions.

All in all, it can be concluded that using a genetic algorithm for the PRESIDENT problem is justified, because the results obtained are good, even for large inputs.

5. Discussion

In Section 3 it was found that the problems PARLIAMENT and PRESIDENT are NP-complete and their optimization versions are inapproximable to a certain extent. These intractability results show that choosing what to vote to favour one's interests is a hard problem, and we think this undermines the capability of democratic voting processes to convert the will of the people into political decisions being consistent with that will in a predictable, transparent, and efficient way. Since it is intractable for voters to just decide what to vote – even when they have a totally clear view of what they actually want in political terms and there is no uncertainty regarding polls and the future behaviour of politicians –, this intractability brings confusion to the process of capturing the actual people's will, making elections obscure and esoteric to some extent. The computational complexity field has been applied before to different social sciences to point out the inefficiency of processes wrongly assumed to be efficient (e.g. the efficiency of markets, in [30]) or to prove the intractability of usual tasks (e.g. the intractability of finding the optimum product mix, in [25]). The intractability results provided in this paper indicate that complexity results can also show the inefficiency of decision making in voting.

Finding optimal solutions for these problems in practice is highly unlikely. For the real case study of the PARLIAMENT problem considered before in Section 4, a brute force exact algorithm would need to explore 1.38e14 different solutions to find the optimal one. Notice that, in the general case, we need to consider $\binom{s+m-1}{m-1}$ different configurations, being s the number of seats and m the number of political parties.[8] Thus, the number of different solutions could be much larger in other cases. For instance, in the European Parliament, where there are 18 main parties and there are 705 seats, the number of solutions increases to $\binom{705+18-1}{18-1} = \binom{722}{17}$. Let us remark that this combinatorial number equals approximately $2.58e34$, which is clearly unfeasible to be solved using any strategy based on brute force.

Regarding the search space of the PRESIDENT problem, the size depends on the number of allies and the number of states. In this case, the number of different solutions would be k^n, being k the number of allies and n the number of states. Thus, for any high enough value for n, it is completely unfeasible to use brute force strategies. For example, for a problem instance with 50 states (like in USA) where the size of groups of allies is 2, the number of different solutions would be 2^{50}, which is unfeasible to handle by exhaustive solution exploration. However, genetic algorithms are good candidates to deal with these situations. In

[8] This is known as the stars and bar problem, and is often use to solve many simple counting problems. One of those problems consist in counting how many different integer configurations solve the equation $x_1 + x_2 + \cdots + x_m = s$, the solution to this problem is $\binom{s+m-1}{m-1}$ different integer configurations [31].

fact, it has been shown that they run fast enough and that they obtain reasonable solutions. Thus, these algorithms can be recommended to deal with these problems.

Let us remark that, in the PARLIAMENT problem, it could be argued that directly distributing the number of seats to form an optimal distribution is impossible for anyone in a democratic parliament. However, given that political polls exist, and that some people may influence the votes of others, it is interesting for them knowing which party should be voted according to the political polls to favour the formation of a parliament which would actually be good for their interests. Moreover, this also illustrates that even the most basic question we could ask to a voter in a parliamentary election, which is what outcome would they prefer, entails high complexity. Thus, deciding what to vote in real parliamentary elections is *at least* that hard.

An interesting aspect to take into account when solving both problems using genetic algorithms is how to deal with the balance between exploration and exploitation during the search for solutions. In any non-trivial problem faced by an evolutionary algorithm, too much exploitation of the most promising areas can lead to abandoning under-explored areas. On the other hand, too much exploration often slows down the process of finding good solutions, leading to searches that are too similar to random search. The balance point between both extremes is difficult to find, depending largely on the type of problem under consideration. In our case, we have seen that for the first problem it is advisable to encourage exploration a little more, while in the second case it is necessary to encourage the exploitation of promising solutions a little more. This fact is evident when analysing different mutation strategies: in the first problem, mutation strategies that allow wider explorations obtain better results; in the second case, more conservative strategies obtain better results.

Finally, we would like to point out that the initial claim of the paper, stating that voting is more difficult than simply choosing the party or candidate whose positions best align with those of the voter, is actually reflected in the results of our experiments. Indeed, in *all* instances of the PARLIAMENT problem we observe that the best configuration of the parliament does not consist in allocating all seats to the party that most closely aligns with the voter's ideas. For example, in the first instance only 4 laws are passed if that "most similar" party gains all seats, but 7 laws are passed by using some alternative solution (which both **mut3** and **mut4** are able to consistently find). The situation is analogous in the PRESIDENT case, where our experiments show that voting just for the target candidate is not the best choice. For example, in the first instance the target candidate can only obtain 108 seats in case all the supporters vote for that candidate. However, 531 representatives can be obtained by asking them to vote for other candidates in some constituencies—in particular, by asking them to vote for certain allies in some constituencies (so that they get representatives who will support the target candidate) and for some non-allies in other constituencies (so that no ally outnumbers the voter's desired candidate overall).

6. Conclusions and future work

Voting may look simple at a first glance. However, after various problems that may arise when voting were analysed, it was observed that it is quite the opposite. Something as simple as deciding which party people should vote for is an NP-complete problem. Despite the completeness of the analysed problems and their inapproximability, it was shown that decent solutions can be found within a reasonable time by using genetic algorithms and properly pruning the solution space. Although obviously people are not expected to run genetic algorithms before going to vote, it is relevant to know that cleverly voting can dramatically affect the outcome of an election. This fact might be interesting for election strategists, and at the same time undermines the capability of elections to transparently capture the will of people, as election outcomes might be very vulnerable to technically-designed political campaigns.

As future work, we are interested in applying other heuristic algorithms to these problems, as well as in studying how these problems are solved in different political configurations. We also wish to study other related problems in the domain of politics, such as measuring the power of a party in terms of the number of majorities it can be part of, and finding mutually beneficial pacts between parties in terms of which laws can be passed.

CRediT authorship contribution statement

Aitor Godoy: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ismael Rodríguez:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Fernando Rubio:** Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The authors would like to thank the anonymous reviewers for valuable suggestions on a previous version of the paper.

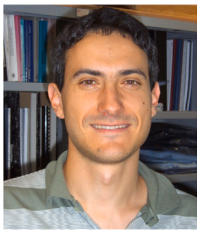
References

[1] S. Merrill, A comparison of efficiency of multicandidate electoral systems, Am. J. Political Sci. 28 (1) (1984) 23–48.
[2] M. Konstantinov, Mathematical aspects of electoral systems, in: AIP Conference Proceedings, Vol. 946, (1) American Institute of Physics, 2007, pp. 55–66.
[3] G. Owen, B. Grofman, Optimal partisan gerrymandering, Polit. Geogr. Q. 7 (1) (1988) 5–22.
[4] S.S.-H. Wang, Three tests for practical evaluation of partisan gerrymandering, Stan. L. Rev. 68 (2016) 1263.
[5] Y. Lewenberg, O. Lev, J.S. Rosenschein, Divide and conquer: Using geographic manipulation to win district-based elections, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 624–632.
[6] E. Eiben, F. Fomin, F. Panolan, K. Simonov, Manipulating districts to win elections: Fine-grained complexity, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, (02) 2020, pp. 1902–1909.
[7] M. Bentert, T. Koana, R. Niedermeier, The complexity of gerrymandering over graphs: Paths and trees, 2021, arXiv preprint arXiv:2102.08905.
[8] J.J. Bartholdi, C.A. Tovey, M.A. Trick, The computational difficulty of manipulating an election, Soc. Choice Welf. 6 (3) (1989) 227–241.
[9] M. Zuckerman, A.D. Procaccia, J.S. Rosenschein, Algorithms for the coalitional manipulation problem, Artificial Intelligence 173 (2) (2009) 392–412.
[10] V. Conitzer, T. Walsh, Barriers to manipulation in voting, in: F. Brandt, V. Conitzer, U. Endriss, J. Lang, A.D. Procaccia (Eds.), Handbook of Computational Social Choice, Cambridge University Press, 2016, pp. 127–145.
[11] O. Keller, A. Hassidim, N. Hazon, New approximations for coalitional manipulation in scoring rules, J. Artificial Intelligence Res. 64 (2019) 109–145.
[12] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998.
[13] S. Katoch, S.S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimedia Tools Appl. 80 (5) (2021) 8091–8126.
[14] D.S. Hochbaum, Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems, in: Approximation Algorithms for NP-Hard Problems, PWS Publishing Co., USA, 1996, pp. 94–143.
[15] M.R. Garey, D.S. Johnson, Computers and Intractability, Vol. 174, Freeman San Francisco, 1979.

[16] M.R. Garey, D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., USA, 1990.
[17] A. Muñoz, F. Rubio, Evaluating genetic algorithms through the approximability hierarchy, J. Comput. Sci. 53 (2021) 101388.
[18] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39.
[19] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.
[20] P. Rabanal, I. Rodriguez, F. Rubio, Applications of river formation dynamics, J. Comput. Sci. 22 (2017) 26–35.
[21] D. Loscos, N. Martí-Oliet, I. Rodríguez, Generalization and completeness of stochastic local search algorithms, Swarm Evol. Comput. 68 (2022) 100982.
[22] B.M. Baker, M. Ayechew, A genetic algorithm for the vehicle routing problem, Comput. Oper. Res. 30 (5) (2003) 787–800.
[23] N.M. Razali, J. Geraghty, et al., Genetic algorithm performance with different selection strategies in solving TSP, in: Proceedings of the World Congress on Engineering, Vol. 2, (1) International Association of Engineers Hong Kong, 2011, pp. 1–6.
[24] I. Rodríguez, F. Rubio, P. Rabanal, Automatic media planning: optimal advertisement placement problems, in: 2016 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2016, pp. 5170–5177.
[25] I. Rodríguez, P. Rabanal, F. Rubio, How to make a best-seller: Optimal product design problems, Appl. Soft Comput. 55 (2017) 178–196.
[26] S. Dash, S. Dey, D. Joshi, G. Trivedi, Minimizing area of VLSI power distribution networks using river formation dynamics, J. Syst. Inf. Technol. (2018).
[27] J.H. Holland, Genetic algorithms and adaptation, in: Adaptive Control of Ill-Defined Systems, Springer, 1984, pp. 317–333.
[28] C.R. Reeves, Genetic algorithms, in: Handbook of Metaheuristics, Springer, 2010, pp. 109–139.
[29] Congreso de los Diputados, Votaciones congreso de los diputados XIV Legislatura, 2023, URL https://www.congreso.es/es/opendata/votaciones. [Internet; Last Access December 2023].
[30] P. Maymin, Markets are efficient if and only if P=NP, Algorithmic Finance 1 (1) (2011) 1–11, Publisher Copyright: © 2011 - IOS Press and the authors..
[31] W. Feller, An Introduction to Probability Theory and Its Applications, Vol. 1, Wiley, 1968, pp. 38–43.

Aitor Godoy is a researcher in the Computer Systems and Computation Department, Complutense University of Madrid (Spain). He obtained a B.S. degree in Mathematics in 2020, and a M.S. degree in Computer Science in 2021. He is currently working on his Ph.D. His research interests cover swarm and evolutionary optimization methods, complexity theory, and the application in the context of social sciences.

Ismael Rodríguez is an Associate Professor in the Computer Systems and Computation Department, Complutense University of Madrid (Spain). He obtained his M.S. degree in Computer Science in 2001 and his Ph.D. in the same subject in 2004. Dr. Rodríguez received the Best Thesis Award of his faculty in 2004. Dr. Rodríguez has published more than 100 papers in international refereed conferences and journals. His research interests cover formal testing techniques, swarm and evolutionary optimization algorithms, computational complexity, formal methods, and functional programming.



Fernando Rubio is an Associate Professor in the Computer Systems and Computation Department, Complutense University of Madrid (Spain). He obtained his M.S. degree in Computer Science in 1997, and he was awarded by the Spanish Ministry of Education with "Primer Premio Nacional Fin de Carrera". He finished his Ph.D. in the same subject four years later. Dr. Rubio received the Best Thesis Award of his faculty in 2001. Dr. Rubio has published more than 100 papers in international refereed conferences and journals. His research interests cover formal methods, swarm and evolutionary optimization methods, parallel computing, and functional programming.