



An efficient heuristic search algorithm for discovering large Condorcet domains

Bei Zhou¹ · Søren Riis²

Received: 13 September 2024 / Revised: 27 December 2024 / Accepted: 28 December 2024 /
Published online: 24 January 2025
© The Author(s) 2025

Abstract

We explore large peak-pit Condorcet domains (CD), an active research area in voting theory. The search for large CDs, defined combinatorially, serves as a benchmark for heuristic-based optimisation algorithms. Since 1996, Fishburn’s alternating scheme produced the largest known CDs for $n \leq 15$ alternatives until recent discoveries surpassed it for $n = 8$ and $n \geq 13$. For $8 < n < 13$, exhaustive searches are infeasible, necessitating the design of heuristic methods. We developed a novel algorithm using a specially designed heuristic function and a 5-alternative subset lookup database to obtain new record-size CDs in this critical range. This approach, distinct from existing methods used for $n \leq 8$, found new large peak-pit CDs: size 1082 (previously 1069) for $n = 10$, and 2349 (previously 2324) for $n = 11$, establishing new lower bounds for these cases. Notably, these new CDs hold restrictions of remarkably small size. These findings fill a significant gap in our knowledge of CDs on $8 < n < 13$ and offer new insights into the structure of large CDs.

Keywords Condorcet domain · Combinatorial optimisation · Machine learning · Heuristic algorithms

Mathematics Subject Classification 03 · 68

1 Introduction

In social choice theory, the study of preference aggregation and decision-making often grapples with majority rule. For each value of n (representing the number of candidates or alternatives), a Condorcet Domain (CD) is a collection of linear orders

✉ Bei Zhou
bei.zhou@imperial.ac.uk

Søren Riis
s.riis@qmul.ac.uk

¹ Imperial College London, London, UK

² Queen Mary University of London, London, UK

(permutations). Crucially, for a CD, every triple of candidates satisfies a specific never condition. This condition, often represented symbolically as xNy , stipulates certain forbidden rankings of candidates. This ensures that cyclical majorities, a situation where no candidate can emerge as a consistent winner in pairwise comparisons, are avoided.

The investigation of CDs, sets of linear orders with an acyclic pairwise majority relation, has been a subject of interest for mathematicians, economists, and mathematical social scientists since the 1950s (Danilov and Koshevoy 2013; Fishburn 2002). CDs also have applications in the Arrovian aggregation and social choice theory (Puppe and Slinko 2024).

In the 1970s and 1980s, researchers initiated studies into the potential magnitude of CDs (Abello and Johnson 1984; Fishburn 1996), leading to a significant breakthrough in 1996 when Fishburn discovered many of the largest CDs, a record that has remained unchallenged for nearly three decades (Fishburn 1996). Since then, researchers have continued the quest to find large CDs (Danilov et al. 2012; Fishburn 1996; Galambos and Reiner 2008; Karpov and Slinko 2023; Monjardet 2009; Leedham-Green et al. 2024), which is closely related to the goal of optimising personal preference freedom while adhering to the requirement of a transitive majority relationship (Raynaud 1982; Puppe and Slinko 2024). In operations search, it is a complex, unsolved combinatorial optimisation problem. Studying CDs has important implications for voting systems and democratic decision-making processes. By identifying large CDs, researchers can gain insights into the properties of voting systems that are more likely to produce outcomes that reflect the electorate's preferences, the strengths and weaknesses of different voting systems, and democratic decision-making processes (Lackner and Lackner 2017). Thus, a key effort in the literature is identifying large Condorcet domains with a paramount objective to uncover the largest CDs (Puppe and Slinko 2024; Karpov 2022; Leedham-Green et al. 2024). To this end, Fishburn (1996) introduced the size function to record the largest CDs on n alternatives

$$f(n) = \max\{|\mathcal{D}| : \mathcal{D} \text{ is a Condorcet domain on a set of } n \text{ alternatives}\}.$$

Additionally, Karpov and Slinko (2023) introduced the following functions:

$$h(n) = \max\{|\mathcal{D}| : \mathcal{D} \text{ is a peak-pit Condorcet domain on a set of } n \text{ alternatives}\}.$$

One important class of large CDs is based on Fishburn's alternating scheme, which involves alternating between two never conditions 2N1 and 2N3 on 3-alternative subsets of candidates (commonly referred to as triples) and has been used to construct many of the largest known CDs on a small number of alternatives. The domains built by the alternating schemes are referred to as Fishburn domains (Slinko 2024) (see Sect. 2 for details). Subsequent research by Galambos and Reiner (2008); Danilov et al. (2012); Monjardet (2009) and Karpov and Slinko (2022) extended and refined this work. Despite extensive work, these Fishburn domains and their design play a central role in constructing large CDs. For instance, the approach introduced in Karpov and Slinko (2023) was built upon the Fishburn domains to construct new CDs of record-breaking CDs for $n \geq 13$, but it does not apply for $n < 13$. Karpov et al.

(2023) introduced the set-alternating schemes that could construct CDs whose size is larger than the Fishburn domains on $n \geq 16$. Their new large CDs also improved the asymptotic lower bound for the size of the largest Condorcet domains (Karpov and Slinko 2023; Karpov et al. 2023).

Another line of studies focuses on developing efficient search algorithms to generate CDs. Akello-Egwel et al. (2024) developed a custom search algorithm that generated all the maximal non-isomorphic CDs on seven alternatives and found the largest CDs on eight alternatives (Leedham-Green et al. 2024) but it does not scale to a larger number of alternatives. Markström et al. (2024) designed a depth-first search algorithm to generate non-isomorphic maximal Arrow's single-peaked domains on 9 alternatives. On a higher number of alternatives, completing their search requires a prohibitively demanding amount of computation. In contrast with these existing search algorithms, our algorithm is universal in the sense that it is applicable to any number of alternatives, and it does not aim to compute a complete set of CDs but focuses on the large ones.

Given the fact that Fishburn domains are not the largest domain on eight alternatives (Leedham-Green et al. 2024) and $n \geq 13$ alternatives (Karpov and Slinko 2023), it is a valid conjecture that for $9 < n < 13$ they are also not the largest. However, existing search algorithms failed to find such CDs. One of our contributions is on the discovery of many CDs larger than the Fishburn domain on $n = 10$ and 11 alternatives. It is worth noting that on $n \geq 9$ alternatives, finding the largest Condorcet domain remains an open problem.

In this paper, we present a heuristic search algorithm for generating and discovering large CDs on $n \geq 6$ alternatives. We first pinpoint the challenges associated with finding large CDs and demonstrate that these challenges pose significant obstacles for common learning algorithms applicable to tackle complex combinatorial optimization problems to find large CDs, especially when dealing with a substantial number of alternatives. This motivated us to develop a heuristic search algorithm tailored to overcome these challenges. Our search algorithm employs an efficient heuristic function that evaluates the goodness of partial CDs based on the size of restriction on 5 alternatives. This idea is grounded on our empirical finding that many locally large restricted CDs also tend to be large, indicating the existence of a linear relationship between the size of a CD and the size of its restriction. Our heuristic function exploits this property and uses a database that contains a complete set of pre-calculated CD restriction information on five alternatives. This idea resonates with the core concept in dynamic programming (Bellman 1966) where a problem is broken into smaller, overlapping subproblems, and instead of solving each subproblem independently, solutions to these subproblems are pre-computed and then stored, so they can be reused when the same subproblem arises.

We further report on new record-breaking CDs on $n = 10$ alternatives (size of 1082) and $n = 11$ alternatives (size of 2349). This discovery has further improved the largest CD since 1996 and opens new directions for research in the theory of CDs, with implications for the theoretical study of voting systems.

The largest domains our search algorithm found on the number of alternatives ranging from 6 to 11, along with the size of their corresponding Fishburn domains,¹

¹ Size of acyclic domain of size n based on the alternating scheme: <https://oeis.org/A144685>.

Table 1 The largest Condorcet domains discovered by our search algorithm on the number of alternatives ranging from 6 to 11

n	Fishburn domain	$f(n)$ and $h(n)$
6	45	45
7	100	100
8	222	<u>224</u>
9	488	≥ 488
10	1069	$\geq \mathbf{1082}$
11	2324	$\geq \mathbf{2349}$

Our work found improved lower bound CDs of size 1082 and 2349 on 10 and 11 alternatives (in bold), respectively and rediscovered the largest CD of size 224 on eight alternatives (underscored)

are presented in Table 1. It is worth noting that along with the new improved lower bound CDs on 10 and 11 alternatives, it also discovered some CDs whose size is larger than those of Fishburn domains for $n = 10$ and $n = 11$ that were the largest known CDs on these number of alternatives.

This paper is organised as follows. In Sect. 2 we provide the preliminary and background knowledge about Condorcet domains from the operations research perspectives, along with defining and explaining the notions and terminologies used throughout this paper. Section 3 illustrates the challenges facing a wide range of existing learning algorithms in finding large CDs and analyses their strengths and weaknesses in addressing this problem. Section 4 introduces a heuristic function and shows its effectiveness in establishing a linear relationship between a CD and restrictions, and further elaborates on integrating it into the heuristic search algorithm. In Sect. 5, we demonstrate the importance of the ordering of triples by which they are assigned and propose a new triple order that contributes partially to the success of the search algorithm. Its detailed configurations, along with an approach that enables it to run on thousands of CPU cores in parallel, are also discussed here. Section 6 contains a detailed analysis of the restrictions of the new large CDs we discovered on 10 and 11 alternatives. We conclude our work in Sect. 7.

2 Preliminaries and background

Let $X_n = [n] = \{1, 2, \dots, n\}$, in which each element in the set is referred to as an *alternative* or candidate in voting theory. A linear order is a set of alternatives with an ordering, where the comparability and transitivity properties hold. Here the comparability property ensures that any pair of alternatives can be compared, and the transitivity property ensures that the ordering of alternatives is consistent. Let $\mathcal{L}(X_n)$ denote the set of all linear orders over X_n . In voting theory, voters have to make a collective choice of which alternative to choose. Each voter can be referred to as an agent. Given N agents, each agent $i \in N$ has a preference order P_i in X_n (each preference order is a linear order and can also be viewed as a *permutation*). For brevity, we represent a preference order as a string, e.g. $12 \dots n$ means 1 is the most favoured

alternative, and n is the least. Note that in this paper, we use preference order, linear order and permutation interchangeably.

A subset of preference orders $\mathcal{D} \subseteq \mathcal{L}(X)$ is called a *domain* of preference orders or permutations. A domain $\mathcal{D}$ is a *Condorcet domain* if whenever the preferences of all agents belong to the domain, the majority relation of the preference profile with an odd number of agents is transitive. Thus a Condorcet domain is essentially a set of permutations where there is always a clear majority preference for any pair of alternatives.

Sen (1966) proved that a domain is a Condorcet domain if and only if restriction of it to each triple of alternatives satisfies some never condition iNj , $i, j \in [3]$. A triple consists of 3 alternatives (x, y, z) where $x < y < z$ and $x, y, z \in [n]$. The never condition iNj dictates that i th alternative from the triple does not fill in j th place within this triple in each order from the domain. Here i takes on the value 1, 2, or 3, representing the smallest, middle, or largest alternatives in the triple, respectively. Similarly, the value of j can be 1, 2, or 3, corresponding to the first, middle, or last position in the triple. For instance, triple (3, 4, 5) on five alternatives with never condition 1N3 dictates that the smallest alternative, 3, cannot present at the last position, forbidding the permutation [4, 2, 5, 1, 3] from presenting in the resulting CD. There are, in total, nine never conditions (1N3, 3N1, 2N1, 2N3, 1N2, 3N2, 1N1, 2N2, 3N3).

A CD's size is defined by the number of permutations contained within it. For any $n \in \mathbb{Z}_{\geq 3}$ alternatives, a CD is associated with a set of $m = \binom{n}{3}$ triples, all of which must be assigned with one of the nine never conditions. A domain is not a CD if there are unassigned triples from which it is created, and we instead call it a *partial CD*. To avoid repetition, we use the acronym TRS (triples and rules²) to refer to a list of triples and their assigned never condition. A TRS produces a CD if all its triples are assigned with a never condition. Otherwise, it is a partial CD. See Zhou et al. (2024) for details on how to construct a CD from a TRS.

A domain which satisfies a never condition of the form $xN3$ for every triple is called *Arrow's single-peaked domain* (Arrow 1963; Raynaud 1981). Slinko (2019) proved that maximal Arrow's single-peaked domain on n alternatives has size 2^{n-1} (see their Theorem 1). A domain $\mathcal{D}$ is a *peak-pit domain* if, for each triple of alternatives, the restriction of the domain to this triple is either single-peaked (satisfying never conditions $iN3$) or single-dipped (satisfying never conditions $iN1$). A Condorcet domain $\mathcal{D}$ is *unitary* if it contains order $123 \dots n$. Since it is without loss of generality to consider unitary CDs, it is sufficient to consider the six never conditions 1N3, 3N1, 2N1, 2N3, 1N2, and 3N2. In the searches, as we focused on discovering peak-pit domains, we narrowed the number of available never conditions down to four (1N3, 3N1, 2N1 and 2N3).

Two unitary CDs are isomorphic if the alternatives can be re-enumerated in a way that they consist of the same set of permutations (Leedham-Green et al. 2024). Given two linear orders P_1 and P_2 , a transformation π is the application of a mapping that converts P_1 to P_2 , denoted as $P_2 = \pi(P_1)$. Given a permutation $P \in D$, a transfor-

² Here rule is a short name for never condition.

mation π_P converts it to its identity form, i.e. $123 \dots n$ and we use $\pi_P(D)$ to denote a domain obtained from converting all the permutations in $\mathcal{D}$ by the transformation π_P .

Any non-empty CD $\mathcal{D}$ is isomorphic to a unitary CD because we can always select a permutation $P \in D$ and then enumerate the alternatives according to the transformation π_P such that the permutation P becomes the identity permutation. Thus, we do not lose any generality by assuming all the CDs we discussed in this paper are unitary.

The *restriction* of a domain $\mathcal{D}$ to a subset $A \subset X$ is a domain with the set of linear orders from $\mathcal{L}(A)$ obtained by restricting each linear order from $\mathcal{D}$ to A (Karpov et al. 2024). Given $k \in [3, n - 1]$ alternatives, a CD on n alternatives can have restrictions on k alternatives by removing the alternatives in every linear order (permutation) within the CD that are not these k alternatives and retaining only one of the duplicated resulting linear orders. The resulting restrictions contain the linear orders that only contain these k alternatives.

Analogously, given $k \in [3, n - 1]$ alternatives, a TRS for n alternatives can also be restricted to k subset triples by retaining the triples that are included in these k alternatives, which is equivalent to removing the triples that include least one of $(n - k)$ alternatives that are not in these k alternatives. The never conditions are transferred from the original triples to the subset triples. A set of triples for n alternative, when restricted to k alternatives subset, produces $\binom{n}{k}$ sets of subset triples, each of which corresponds to one combination of k alternatives from n alternatives. For example, a set of 11 alternatives produces 10 sets of subset triples when restricted to 10 alternative subsets and 55 sets of subsets when restricted to 9 alternative subsets.

Before our results, the largest known CDs on 10 and 11 alternatives were Fishburn domains. A domain $\mathcal{D}$ is called *Fishburn domain* if it satisfies the alternating scheme (Fishburn 1996): there exists a linear ordering of alternatives $a_1, \dots, a_n$ such that for all i, j, k with $1 \leq i < j < k \leq n$ the restriction of the domain to the set $\{a_i, a_j, a_k\}$ is single-peaked if j is even, and it is single-dipped if j is odd. When applying the alternating scheme to the list of triples (x_i, y_i, z_i) , it dictates that if the middle alternative y_i in a triple is even, it is assigned with never condition $2N3$, otherwise $2N1$. Despite its simplicity, it turned out that the list of generated never conditions built large CDs that had held the records for almost three decades until recent studies discovered some new large CDs on eight alternatives (Leedham-Green et al. 2024) and $n \geq 13$ alternatives (Karpov and Slinko 2023).

The first complete list of all maximal CDs on five alternatives has been obtained by Dittrich (2018). CDs on five alternatives are constructed on 10 triples, as demonstrated in Table 2, where their never conditions are assigned by applying the alternating scheme. The resulting Fishburn domain on five alternatives has size 20. The permutations inside the CD are presented in Table 3. It is the largest CD on five alternatives.

The Fishburn domain on five alternatives has five restrictions on four alternatives, whose sizes are 9, 9, 8, 9, 9 as demonstrated in Table 4 where each restriction has one alternative removed from the original CD. For example, the restriction on four alternatives produced by removing alternative one from these 20 permutations in this Fishburn domain contains nine permutations.

There is a positive correlation between the size of a CD and the size of its restrictions (Karpov et al. 2024), and to illustrate we use the lattice structure to show the size of a

Table 2 The list of triples for five alternatives and their never conditions are assigned according to the alternating scheme

Triples	Never conditions
(1, 2, 3)	2N3
(1, 2, 4)	2N3
(1, 2, 5)	2N3
(1, 3, 4)	2N1
(1, 3, 5)	2N1
(1, 4, 5)	2N3
(2, 3, 4)	2N1
(2, 3, 5)	2N1
(2, 4, 5)	2N3
(3, 4, 5)	2N3

Table 3 The Fishburn domain on five alternatives

Fishburn domain on five alternatives			
12,453	12,435	12,345	54,321
45,321	54,231	45,231	42,531
24,531	54,213	45,213	42,513
42,153	42,135	24,513	24,153
24,135	21,453	21,435	21,345

It consists of 20 permutations built from the triples and never conditions in Table 2

Table 4 The four alternative restrictions from the Fishburn domain on five alternatives

Removed alternative	Restrictions					
1	4253	4523	5432	2345	5423	
	2453	4532	4235	2435		
2	4531	5413	4513	5431	4135	
	1453	4153	1345	1435		
3	4521	4215	2145	2415	1245	
	4251	5421	2451			
4	1235	2513	2153	2135	2531	
	1253	5213	5231	5321		
5	2143	2134	1234	2431	4213	
	4231	2413	1243	4321		

Fishburn domain on five alternatives and the sizes of their restrictions when restricted to 4 alternatives in Fig. 1. The number sitting aside the arrows is an alternative and indicates that the restriction is built by eliminating that alternative from permutations in the original CD. The largest CD on four alternatives is of size 9. Four restrictions are of the maximum size for four alternatives, and only one CD is of size 8. To make a comparison, we also show in Fig. 2 a CD on five alternatives of size 17 and its restriction sizes on 4 alternatives. Two of the restrictions are of size 7. It is evident that,

Fig. 1 The restriction sizes for the Fishburn domain on five alternatives when restricted to four alternatives

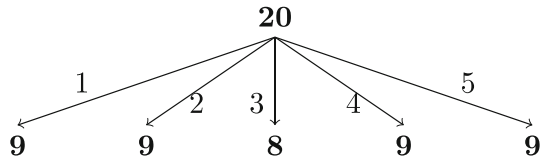
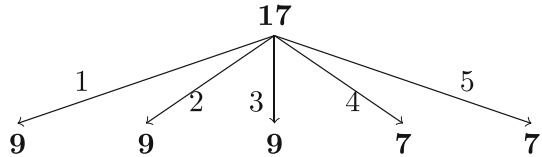


Fig. 2 The size of five restrictions on four alternatives from a five alternative CD of size 17



in this case, a larger CD leads to a set of larger restrictions. We further discovered that this monotonicity property holds for many CDs in general, though not all larger CDs induce larger restrictions. Based on this finding, we constructed a heuristic function that captures this correlation. We show in Sect. 4 that this heuristic function establishes a nearly linear relationship between the size of a CD and a value calculated from an aggregation of its five alternative restriction sizes.

3 Suboptimality of existing learning algorithms

The search for large CDs is a challenging discrete combinatorial optimisation problem that requires significant effort and faces several challenges where many common learning algorithms struggle. While learning algorithms like reinforcement learning algorithms, genetic algorithms, and local search algorithms have proven their effectiveness in solving some combinatorial optimisation problems (Fawzi et al. 2022; Mankowitz et al. 2023; Radhakrishnan and Jeyakumar 2021; El-Shorbagy and Hassanien 2018; Lauri et al. 2023), they encounter immense difficulty in finding large CDs for $n \geq 8$ alternatives.

We tested a range of algorithms that can solve discrete combinatorial optimisation problems such as finding large CDs. All the programs we used are publicly available.³ To ensure a fair comparison, all algorithms were run on a single CPU core and terminated after 24 h if no improvement was observed. We set up these experiments in this way because we compared many algorithms each of which has unique characteristics, it is unfair to compare them given the same amount of running time. Instead, we ran them to their full potential and also gave them more time to ensure they had reached their full capacity. The large CDs they found on four to eight alternatives are presented in Table 5. Apart from the simple hill climbing algorithm, all the tested learning algorithms found the largest CD on four and five alternatives. But on seven and eight alternatives, none of them found the largest known CDs. This indicates that while these algorithms can find large CDs, there are challenges for them in finding higher lower bounds for CDs on a higher number of alternative alternatives. We summarize these challenges as follows.

³ GitHub <https://github.com/sagebei/cdl/tree/main/algorithms/benchmarking>.

The most prominent difficulty facing them is the exponential growth of the search space. As the number of alternatives increases, the number of potential solutions expands exponentially, making exhaustive exploration impractical or computationally infeasible for $n \geq 6$ (Leedham-Green et al. 2024). To overcome this issue, prior works have designed search algorithms capable of significantly reducing the search space. Akello-Egwel et al. (2024) released complete sets of isomorphic CDs on $4 \leq n \leq 7$ alternatives, which were obtained from a search algorithm they developed that produces all the maximum CDs on a given number of alternatives whose size is larger than or equal to a cutoff value. Using a paralleled version of this algorithm on a supercomputer, Leedham-Green et al. (2024) found the largest CDs on eight alternatives. However, their algorithms are not practically applicable to finding the maximal CD for $n \geq 9$ as the search tree that the algorithm needs to build is too large. For a full peak-pit CD of 9 alternatives, there are 84 triples, and thus, when each of them is assigned a never condition chosen from a list of 4 never conditions, it results in a search space of size 4^{84} . In practice, though their algorithm can reduce the search space, it is still impossible to search through the reduced search space exhaustively. Additionally, although many structures in the search space could be exploited (Puppe and Slinko 2024), greedy strategies that reduce the search space by narrowing the search to focus solely on large partial CDs failed miserably.

Consequently, these algorithms often use approximate methods to navigate the vast solution space efficiently. However, the intricate interdependencies among these triples exacerbate the problem. The presence of complex relationships significantly restricts the feasible search space, making it challenging for algorithms to explore all potential solutions effectively. A mistake at any step is likely to spoil the whole solution.

The existence of many local optima corresponding to suboptimal solutions poses another significant hurdle for these algorithms. Local search algorithms and evolutionary techniques, including genetic algorithms, are particularly prone to getting trapped in local maxima due to their reliance on iterative refinement processes. Escaping these local optima and effectively exploring the entire solution space to converge to the global optimum is a challenging task. From a general understanding of the complete set of CDs on seven alternatives (Akello-Egwel et al. 2024), many large domains have sizes that are close to the maximum possible, but they represent different local optima. Thus, many learning algorithms get trapped in the swamp of the large domains, making it hard to escape and find the largest domain.

Calculating the size of the CD on a large number of alternatives is computationally heavy and time-consuming (Zhou et al. 2024), leading to another major issue where it is computationally demanding to compute the resulting CD sizes millions of times needed for the learning algorithms to improve on the feedback.

Finding large CDs can be naturally cast as a reinforcement learning problem where it can be modelled as a single-player game. The game starts with an initial position, a list of empty triples with no never condition assigned. In each game step, the player assigns a never condition to the next triple in the sequence. It ends when all the triples are assigned. There are no intermediate rewards, and the final reward is in proportion to the size of the constructed Condorcet game at the end of the game. In this setting, an RL agent sequentially assigns a never condition to the list of ordered triples and gets a reward.

Table 5 The largest CDs found by a range of learning algorithms

n	Largest size	Deep RL			EC			Local search	
		DQN	PPO	A2C	GA	ES	PSO	HC	SA
4	9	9	9	9	9	9	9	9	9
5	20	20	20	20	20	20	20	16	20
6	45	44	45	41	44	41	45	41	45
7	100	86	94	85	96	81	92	71	97
8	224	180	199	181	194	162	182	145	216

Common learning algorithms are vulnerable to local traps and victims of huge search space in finding large CDs

DQN, Deep Q learning; PPO, Proximal Policy Optimisation; A2C, Advantage Actor-Critic; EC, Evolutionary Computation; GA, Genetic algorithms; ES, Evolutionary strategies; PSO, Particle Swarm Optimisation; HC, Hill climbing; SA, Simulated annealing

Deep reinforcement learning (DRL) algorithms, while powerful, still have limitations when it comes to solving hard combinatorial optimisation problems (Mazyavkina et al. 2021; Barrett et al. 2020). Deep Q learning algorithms (Mnih et al. 2015), Proximal Policy Optimisation algorithms (Schulman et al. 2017) and Advantage Actor-Critic algorithms (Mnih et al. 2016) in their original form are prone to converge to local optima as Barrett et al. (2020) described. In our experiments, they all failed at finding the largest CDs on $n \geq 7$ alternatives.

Given these identified challenges, we analyze the following factors contributing to this issue. One of the main limitations is the curse of dimensionality, which refers to the exponential increase in the number of possible solutions as the problem size grows, which is the exact problem facing finding the largest CDs. This can make it difficult for DRL algorithms to search through all possible solutions and find the optimal one in a reasonable amount of time. Another limitation is the presence of local optima, which traps the algorithm in a sub-optimal solution. This is particularly true in our case where the reward function is sparse (i.e. a reward is given only at the end of the game), making it difficult to differentiate between good and bad solutions. While the size of the partial CDs is a good choice for intermediate rewards, they do not strongly correlate with the size of the resulting full CD and sometimes can be misleading. DRL algorithms also require a lot of computational power, memory, and data to train effectively, which can be a challenge for working with large alternative CDs. Additionally, finding the set of never conditions that construct a large CD requires a high degree of precision, which is difficult to achieve with reinforcement learning algorithms.

The AlphaZero-style algorithms, one of the deep reinforcement learning algorithms, has achieved remarkable success on board games like Chess and Go by utilizing a value function⁴ that evaluates the current board position without waiting for the game's completion, as described by Silver et al. (2018). The AlphaZero algorithm and its variations do not only apply to board games but are also applicable to solving combinatorial optimisation problems (Laterre et al. 2018). For example, Fawzi et al.

⁴ Note that AlphaZero algorithms also use policy function to cut off poor branches, but it is out of the scope of this paper to discuss the details.

(2022) applied a variation of AlphaZero called AlphaTensor to improve the best-known matrix multiplication algorithms, and Mankowitz et al. (2023) developed AlphaDev that found fast sorting algorithms, further showcasing the algorithm's potential on tackling hard combinatorial optimisation problems.

However, training or building an accurate optimal value function is challenging. A recent study has shown that when the estimations of the value function are not accurate, i.e., it fails to predict the precise optimal value of states, the direction of the search could be misled to the points where the algorithm fails to learn to play the child game of Nim, one of the classic impartial games (Zhou and Riis 2022). That, along with some of the literature it referred to, also shows that modelling parity function by neural networks on long bitstrings is a non-trivial task. To some extent, the parity issue that characterizes the challenge of learning Nim-like games also arises in CDs. Their study also highlights the difficulty of modelling the parity function using neural networks on long bitstrings, which characterizes the challenges of learning Nim-like games and CDs. As discussed in Sect. 2 all previously known CDs have utilized the alternating scheme, which explicitly employs the parity function.

We experimented with two local search algorithms, random-restart hill climbing and simulated annealing with restarts. As expected, they struggled. On four, five, and six alternatives, the latter found the maximal size of 9, 20, and 45 (both when searching with the four pit-peak never conditions and all six never conditions). For seven alternatives, the largest CD size it found was 97, and for eight alternatives, it found a CD of size 216. A major issue is that it is computationally demanding to compute CD sizes the millions of times needed for the algorithm to get enough feedback to efficiently explore the search space and escape from the local trap to reach the global optima.

Since simulated annealing proved to be the most efficient algorithm for this task, we thus used it to find large CDs for 9 and 10 alternatives. After running for 24 h, the sizes of the largest CDs it found were 455 and 937 for 9 and 10 alternatives, respectively. While it was not feasible to test all parameter combinations, we selected reasonable ones and refined them when the results were unpromising. Despite these efforts, it did not find any CDs larger than the corresponding Fishburn domains.

The fact that the genetic algorithms and simulated annealing found relatively larger CDs for seven and eight alternatives in comparison with other algorithms shows that focusing on a small population of good solutions and further exploring them is a good strategy, and on the contrary, striving to build a good solution from scratch tends to end up in local traps. The utility of value functions in RL algorithms in solving mathematical problems (Fawzi et al. 2022) and combinatorial optimisation problems (Mazyavkina et al. 2021), along with the benefit of focusing on a set of good candidate solutions as in GA and SA inspired our heuristic search method in which we designed a handcrafted heuristic function to calculate a value estimating the goodness of a set of partially assigned triples, and the search further explores the ones that have high evaluations.

4 The algorithm

Using the partial CD size as a measure to evaluate it leads to two issues. Calculating the size of a partial CD on a large number of alternatives entails prohibitively costly computation. More importantly, there is no guarantee that a large partial CD could lead to a large full CD, which is why the greedy algorithms stumble, indicating that using the size of a partial CD as a measurement to rank them is erroneous. This promotes the need to develop heuristic functions to evaluate them. A best heuristic function establishes a strong linear relationship between the size of a CD and its value for all the CDs where for any two CDs, $\mathcal{D}_1$ and $\mathcal{D}_2$, if the size of $\mathcal{D}_1$ is larger than that of $\mathcal{D}_2$, the value of $\mathcal{D}_1$ is also higher than that of $\mathcal{D}_2$. However, training or building a best heuristic function is challenging. In this section, we present a search algorithm that applies a manually designed heuristic function using a database that could capture the positive relationship.

4.1 Database construction

Our empirical observation that the size of a CD and the sizes of its restrictions are linearly correlated suggests that the sizes of the restrictions of two CDs can be utilised to measure their relative size. To extrapolate, the largest full CD size reachable from a partial CD is also correlated with the sizes of the largest full restrictions reachable from its restrictions. On a small number of alternatives, we designed an efficient algorithm to find out the size of the largest CD any partial CD can end up with, and store them in a database to enable fast look-up.

The database contains a set of precomputed (key : value) pairs where the keys are the states of fully and partially assigned triples for all combinations of never condition assignment on five alternatives and their corresponding value is the largest possible full CD size obtainable from a partial CD or the actual CD size for a full CD. The TRS state conception was proposed in Zhou et al. (2024), which refers to the string representation of the never conditions assigned to a list of triples in an order. This database enables swiftly obtaining the largest possible size of any five alternative CDs given their state by simply looking it up. An efficient algorithm that constructs the database on $3 \leq n \leq 5$ is given in Algorithm 1. Considering the exponentially growing computation complexity of building the database as n grows larger and the memory needed to store the dataset, it is impractical to construct it on $n \geq 6$ alternatives.

4.2 The heuristic function

The main objective of a heuristic function is evaluating partial CDs. For a partial CD, the heuristic function should calculate a value in proportion to the largest possible full CD size obtainable from it. That is, for any given CD, the heuristic function should establish a trend in which the larger the full CD that a partial CD can be, the larger its value. Given that all the CDs for $n \leq 7$ have been studied, we aim to develop the heuristic function that applies to the CDs on $n \geq 8$ alternatives.

Algorithm 1 Construct the database

Input: The number of alternatives n where $n \geq 3$
 The total number of triples $m = \binom{n}{3}$
 A set of never conditions R : 2N1, 2N3, 1N3, 3N1 by default

Output: A key-value database $\mathcal{D}$ on n alternatives

Add $\|R\|^m$ (4^m by default) pairs of (state of full TRS with R : their resulting CD size) to $\mathcal{D}$

for $i = 1, 2, \dots, m - 1$ **do**

$j \leftarrow m - i$
 $k \leftarrow \binom{m}{j}$
 $\mathbf{R}_j \leftarrow$ Cartesian product of $\underbrace{R \times R \times \dots \times R}_j$

TRS $\leftarrow$ a list of k TRSs each with j triples assigned with 3N1

for each TRS in **TRS** **do**

$T_i \leftarrow$ unassigned triples in TRS
 $T_j \leftarrow$ assigned triples in TRS
for each R_j in $\mathbf{R}_j$ **do**
 Reassign R_j to T_j for TRS
 largest_size $\leftarrow 0$
for each unassigned triple t in T_i **do**
for each never condition r in R **do**
 Assign r to t for TRS
 size $\leftarrow$ look up TRS state in $\mathcal{D}$
if size > largest_size **then:**
 | largest_size $\leftarrow$ size
 Remove the never condition for t in the TRS $\triangleright$ leave t unassigned

Add (TRS state : largest_size) pair to $\mathcal{D}$

For any TRS on n alternatives where $n \geq 6$, the heuristic function given in Equation (1) takes as input a complete set of $s(= \binom{n}{5})$ subset states and look them up in the database to fetch their values $\mathbf{v} = [v_1, v_2, \dots, v_s]$ where $v_i \in [1, 20]$. They are then aggregated to obtain a single value estimating the goodness of this TRS. The aggregation is a linear weighted combination of the number of occurrences of the size of each of the five alternative CDs.

$$f(\mathbf{w}, \mathbf{v}) = \begin{cases} -1 & \text{if } \min(\mathbf{v}) \leq 15 \\ \sum_{i=16}^{20} m_i \cdot w_i & \text{otherwise} \end{cases} \quad (1)$$

where m_i denotes the number of occurrences for five alternative CDs of size i , and w_i is its corresponding weight. To lead the search to find the largest CDs possible, the heuristic function does not account for the 5 alternative restrictions whose size is less

than 16 because the size of the 5 alternatives restrictions on the largest CDs on $n \leq 8$ alternatives is larger than or equal to 16 Karpov et al. (2024); Leedham-Green et al. (2024). We conjecture that this also applied for $n > 8$. Considering that the larger CDs should, in general, contain higher numbers of larger restrictions, we set the weights with increment values [1, 2, 3, 4, 5], i.e. $w_{16} = 1$, $w_{17} = 2$, $w_{18} = 3$, $w_{19} = 4$, $w_{20} = 5$.

Akello-Egwel et al. (2024) released two complete sets of unitary non-isomorphic CDs for six and seven alternatives.⁵ These CDs enabled us to evaluate the heuristic functions by examining the relationship they give rise to between their sizes and their values. Using these CDs, we show in Fig. 3 two scatter plots for six and seven alternatives, demonstrating the relationship between the size of the CDs and their values from our heuristic function. It is conspicuous that for $n = 6$ CDs whose size is larger than 28 and for $n = 7$ CDs whose size is larger than 44, the heuristic function builds a linear relationship between the size and the values of CDs. The fact that the correlation is not strong on small CDs is not problematic as this issue can be resolved by the prioritisation strategy employed in the search algorithm that could rule them out. Overall, our heuristic function, though not perfect, serves its purpose well.

4.3 The heuristic search algorithm

In the selection phase of genetic algorithms, solutions in the population are evaluated using a fitness function, which measures how well they solve the problem and the ones that perform have higher fitness are more likely to be selected for the next generation. Owing to its effectiveness, we employ the same strategy in ruling out inferior TRSs that are not likely to end up producing a large CD.

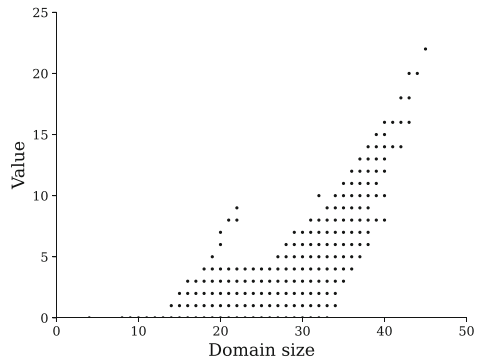
An exhaustive search algorithm that builds an entire search tree begins with a root node corresponding to an empty TRS where the triples are all unassigned. The root node is expanded by adding four child nodes, representing 4 TRSs where the first triple is assigned to one of the four never conditions, 2N1, 2N3, 1N3, and 3N1. Each child node will then be expanded, and this process continues iteratively until all the triples are assigned. To avoid building an enormous search tree, at each search step our heuristic search algorithm keeps N high-ranking nodes for the next step. The heuristic search algorithm that uses the heuristic function with access to the database is given in Algorithm 2.

Choosing a larger value of N may not always lead to better search results. In fact, due to the nonmonotonicity of the heuristic function, increasing N with the same heuristic function can sometimes result in worse search results. This happens because, at the early stage of the search, many TRSs with higher values that lead to smaller CDs outcompete the ones with lower values that could construct larger CDs. Thus, retaining more TRSs could potentially introduce more inferior TRSs.

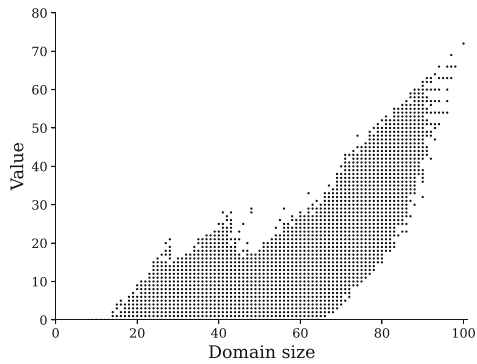
In such cases, adjusting the parameters in the heuristic function to build a stronger correlation may be necessary. However, our aim is to provide a new paradigm to search for large CDs and we do not presume the heuristic function we developed is optimal. To find better parameters, simply testing out various weight combinations

⁵ Condorcet Domains: <http://abel.math.umu.se/~klasm/Data/CONDORCET/>.

Fig. 3 The correlation between the sizes of domains and their values evaluated by the value function for six and seven alternative CDs. Plot **a** includes all the six alternative isomorphic CDs; plot **b** uses 100,000 randomly sampled $n = 7$ CDs for each size since the number of CDs is too many



(a) The values of all $n=6$ non-isomorphic CDs, based on heuristic function (1)



(b) The values of some $n=7$ non-isomorphic CDs for each size between 4 and 100, based on heuristic function (1)

does not help significantly as our experiences indicate. We instead propose to utilise machine learning techniques to find better parameters using the data in Akello-Egwel et al. (2024). Another example suggesting the non-monotonic property is that the search algorithm using a heuristic function calculating the value from restrictions on 6 alternatives did not find any larger CDs than the Fishburn domains. Here, as it is prohibitively impractical to build the dataset for 6 alternatives, we computed the largest possible restriction sizes using the process described in Algorithm 1.

Searching for large CDs entails a significant amount of computation, especially on CDs with many alternatives. One of the advantages of our algorithms is that the computation needed to evaluate each partial CD by the value function is much less than calculating its actual size, and the overhead of working with a larger number of alternatives only increases slightly. While our search algorithms have discovered new unknown large CDs for 10 and 11 alternatives, we conjecture there are larger CDs for $n = 9$ that our search algorithms have not found, indicating that there is still some room for further improvement. The heuristic function we developed, based on the 5

Algorithm 2 The heuristic search algorithm

Input: The number of alternatives n where $n \geq 6$
 A set of never conditions R : 2N1, 2N3, 1N3, 3N1 by default
 The number of top TRSs kept in the search N where $N \geq 1$
 The heuristic function f

Output: $\mathbf{TRS}_N$ (N full TRSs with the highest values)

Create an empty list $\mathbf{TRS}_N$
 Initiate a TRS where all the triples are unassigned and add it to $\mathbf{TRS}_N$
 $T \leftarrow$ a complete list of $m = \binom{n}{3}$ triples
for each triple t in T **do**
 for each TRS in $\mathbf{TRS}_N$ **do**
 for each never condition r in R **do**
 Assign r to t for TRS
 Add TRS to $\mathbf{TRS}_N$
 Remove the never condition for t in the TRS
 Calculate the value for every TRS in $\mathbf{TRS}_N$ using heuristic function f
 Remove the TRSs from $\mathbf{TRS}_N$ on which the heuristic function returns -1
 $\mathbf{TRS}_N \leftarrow$ Top N TRSs with the highest values

alternative restriction sizes, is practically accurate, but it is by no means perfect. Its success shows a strong correlation between the actual CD size and its corresponding subset sizes, and exploiting this feature in more innovative ways opens a door for future research. At least, given the current construction, many other weight combinations can also be tested out.

5 Experiments and results

All our experiments were built on top of the Condorcet Domain Library (CDL) (Zhou et al. 2024) which provides a wide range of functionalities for CD-related calculations. This library was initially developed to support this work. It has underpinned some research studies (Karpov et al. 2023; Akello-Egwel et al. 2024; Karpov et al. 2024; Markström et al. 2024) since its initial release due to its flexibility and fast execution speed. The fast computation enabled by this library contributes significantly to the success of our algorithm. Especially, the `subset_states` function it provided, which converts a TRS to its subset states for $n \geq 6$ alternatives, facilitated the implementation of our search algorithms.

Our search algorithm found the largest CD for $n = 6$ within a second, as shown in Table 6. As the number of alternatives increased, the time it took to find a large CD skyrocketed not only because the search space was growing exponentially but because more high-value nodes were needed to participate in the search to get good results. It

Table 6 The time consumption of our method on rediscovering the known largest CD for $n \leq 9$ on a Macbook Air with 1.1 GHz Dual-Core Intel Core i3 and 8 Gigabyte memory

n	Size of largest CDs found	Time consumption
6	45	0.605 s
7	100	1.31 s
8	222	93 s
9	488	31 min

is worth noting that this heuristic search algorithm has been integrated into the CDL library. The codes are available at the CDL GitHub repository.⁶

On more than nine alternatives, the order of the triples by which they are assigned with a never condition is crucial and has a non-negligible impact on the CDs found by our search algorithm. A well-chosen ordering helps cut off unfruitful branches at the early search stage. To verify this, we developed an approach that selects the next triple to be assigned a never condition dynamically, aiming to find the triple among all the unassigned ones that reduce the size of the CD the most once assigned. This approach records the size of a partial CD for each unassigned triple when assigned with one of the 4 never conditions that maximize the size of this partial CD and choose the one with the lowest value. Dynamically selecting the next triple gives good results, but it requires a large amount of calculation to access all the unassigned triples, and it is prohibitively expensive to employ this strategy when working on $n \geq 8$ alternatives.

We experimented with different static triple orders where the triples were listed randomly or sorted, and the order was fixed throughout the search once determined. One of our special static orders based on various mathematical heuristics and proprieties stood out and led to the discovery of record-breaking CDs. The ordering of the triples we used is a type of static ordering and is defined such that triple (x_1, y_1, z_1) is before (x_2, y_2, z_2) if $x_1 < x_2$ or $(x_1 = x_2 \text{ and } z_1 < z_2)$ or $(x_1 = x_2 \text{ and } z_1 = z_2 \text{ and } y_1 < y_2)$. This choice of triple order is a contributing factor that enables our method to find the existing large CDs constructed by alternating schemes quickly. We call this ordering RZ-ordering, where RZ is the authors' initials.

One major benefit brought by RZ-order is that the order of the list of triples in a TRS on $n \geq 6$ alternatives when restricted to five alternatives remains unchanged, which is to say that when the triples in the five alternatives database are in RZ-order, for $n \geq 6$ alternatives the restricted triples to five alternatives are also in RZ-order, meaning that they can be looked up in that database without reordering. However, for other orders of triples that do not hold this property, like the lexicographic order, many of the orders of triples restricted to five alternatives are not in lexicographic order. They have to be reordered before being able to be looked up in the dataset where triples are also in lexicographic order, incurring extra cost of computation.

In the experiments where the new CDs are found, we kept the top $N = 200000$ and $N = 100000$ high-value nodes for 10 and 11 alternatives, respectively. These experiments were conducted on high-performance machines, but we did not record the exact execution times. Based on our estimates, for $n = 10$, the computations took approximately 25h, while for $n = 11$, they required around 100h. Table 7 displays

⁶ GitHub <https://github.com/sagebei/cdl/tree/main/python>.

Table 7 The sizes and counts of new large CDs on 10 and 11 alternatives, discovered by the heuristic search algorithm

n	New large CD sizes and their count							
10	Size	1070	1071	1072	1074	1078	1079	1082
	Count	80	29	70	37	13	8	5
11	Size	2328	2329	2334	2337	2349		
	Count	330	16	8	67	122		

the size and their counts of the CDs that are larger than the Fishburn domains which are the largest known CDs on 10 and 11 alternatives. The largest CDs we found on 10 alternatives have size 1082, while the size of other smaller CDs ranges from 1070 to 1079. On 11 alternatives, there are CDs of 5 different sizes larger than the Fishburn domain discovered by the search algorithm, with the largest one being 2349.

It is sensible to claim that the larger the size of CDs, the harder they are to be discovered. However, our method found more (isomorphic) CDs of size 1072 than those of size 1071 for 10 alternatives and more CDs of size 2337 than those of size 2334 for 11 alternatives. This reflects one of the challenges mentioned above in the search where designing a successful heuristic function that prioritizes the large CD is challenging. It also insinuates that the heuristic function we crafted, although with significant efforts, is imperfect. We see it as an open research problem to design an efficient and effective heuristic function that distinguishes the CDs of different sizes with a large margin such that the unpromising partial CDs can be ruled out with high confidence.

This search algorithm, though relying on access to a database, can be scaled to run on thousands of CPU cores on parallel, each of which independently starts a search on a partial CD. We experimented with this technique by starting 1000 searches on 1000 CPU cores on QMUL HPC machine (King et al. 2021), and found CDs whose size is 224 for eight alternatives, the largest CD for $n = 8$ (Leedham-Green et al. 2024).

The whole search was split into two stages. In the first stage, the search began with a TRS with no never condition assigned and stopped after a given triple (the 15th triple in our experiment) in TRSs was assigned, resulting in a large collection of TRSs where the top triples above that triple (the first 15 triples in our case) were all assigned. The first stage ended with evenly splitting the list of TRSs into 1000 chunks. They were shuffled before splitting to ensure that the TRSs in one chunk were not similar. In the second stage, we resumed the search on 1000 CPU cores with the N , the number of high-value CDs kept being 10,000, each of which searched for a chunk independently. When they were finished, the size of all the CDs found by the search was calculated, and we found 17 CDs of size 224.

Paralleling the search algorithm in this way undoubtedly accelerates the computation by engaging in many CPU cores, but it has one drawback. As the list of TRSs is segregated into isolated blocks, and the search is conducted within each block, each block is only aware of the values of the TRSs in itself.

Table 8 The 10 alternative restriction sizes of these $n = 11$ CDs larger than the Fishburn domain

$n = 11$ CD sizes	Restriction sizes on 10 alternatives									
2328	1003	1005	1022	1043	1053	1072	1074	1079		
2329	1013	1028	1039	1040	1041	1044	1057	1058	1063	1079
2334	1021	1030	1035	1039	1043	1057	1063	1064	1078	
2337	1017	1022	1031	1045	1053	1062	1072	1074		
2349	1021	1026	1035	1045	1053	1068	1074	1078		

Each contains at least one $n = 10$ CD larger than the Fishburn domain when restricted to some 10 alternatives. These CD sizes are highlighted in bold

We devoted considerable effort to obtaining new record CDs for $n = 12$ and $n = 13$. However, our progress was constrained by the limited computational resources available. The core of our algorithm requires storing a substantial number of TRSs in RAM, and identifying larger CDs for higher n values demands an even greater volume of TRSs, significantly increasing memory usage. Furthermore, the computational time required grows exponentially with the number of alternatives, presenting an additional challenge.

6 Analysis of discovered new large CDs

The most commonly discussed properties of CDs, such as their width and radius, etc., have been calculated for these new large CDs and are provided in the supplementary material. Besides that, as the restrictions of a CD reveal its inner local structure and their application plays a central role in the heuristic function employed in the search algorithm, we use the size of restrictions and their counts as a measure of the CD structure. On 11 alternatives, the search algorithms found five CDs of different sizes larger than the Fishburn domain. Table 8 shows that each of these large CDs when restricted to 10 alternative subsets, produces at least one restriction larger than the Fishburn domain. Still, they were all found by our method, indicating our method is consistent across different numbers of alternatives. It is also worth pointing out that none of the restrictions is identical to the Fishburn domain, showing that the inner structure of these new CDs differs fundamentally from that of the Fishburn domain.

On 10 and 11 alternatives, the Fishburn domains had the largest size before our results. Here we compare the largest CDs found by the search algorithms with them. We calculated the restriction sizes for the largest CD we found on $n = 10$ and 11 alternatives when restricted to a range of 4 to $n - 1$ alternatives. Their sizes and counts are listed in Tables 9 and 10 for 10 and 11 alternatives respectively. We observed that at least one of the restrictions on the 10 alternative Fishburn domain is also a Fishburn domain. However, for the largest $n = 10$ CD our algorithm found, the size of its largest restriction on nine alternatives is 485, which is smaller than the $n = 9$ Fishburn domain. We thus speculate that it is not the largest CD on 10 alternatives. It provides an improved low bound, but there is no evidence showing that it is the high bound on the domain size.

Table 9 The restriction sizes and their count for the 10 alternative CD of size 1082 when restricted to 4 to 9 alternative subsets

Subset n	Restriction sizes and their count											
4	Size	8	9									
	Count	90	120									
5	Size	16	17	18	19	20						
	Count	6	30	6	162	48						
6	Size	36	39	40	41	42	43	44	45			
	Count	6	6	2	44	86	40	6	20			
7	Size	87	89	91	92	93	94	95	96	97	98	100
	Count	6	2	4	16	22	18	8	16	6	16	6
8	Size	200	204	205	209	211	212	214	216	218	219	222
	Count	2	4	4	4	3	2	7	10	4	4	1
9	Size	473	481	485								
	Count	4	2	4								

Furthermore, the restrictions from Fishburn domains generally have a larger number of large CDs than the ones from the CD of size 1082. For example, when restricted to eight alternatives, only one restriction is of size 222, in contrast to the ones from the Fishburn domain, which contain 15 restrictions whose size is 222. This shows that larger CDs do not always produce larger restrictions or a higher number of larger restrictions compared to those from smaller CDs. This aligns with the conclusion in Karpov et al. (2024) that suggests CDs that have the largest restriction might not be the largest.

We observed a similar pattern with the restrictions on the 11 alternative CD of size 2349. As shown in Table 10, some of these restrictions on the 5, 6, and 7 alternatives, have the same size as the the Fishburn domains. But it also has relatively small restrictions, especially on seven alternatives where the size of the smallest restriction is 74. All the restrictions on 9 alternatives are smaller than Fishburn domains.

Our findings here show that on this $n = 10$ CD, there exist restrictions on 4 and 5 alternatives with size 8 and 16, respectively, meaning that for k values of 4 and 5, n value of 10 already satisfies the condition. On the 11 alternatives CD of size 2349, the condition for $k = 6$ is satisfied on the largest CD we discovered on $n = 11$ alternatives where some of the restrictions on 6 alternatives have size 32.

7 Conclusion

In this paper, we present a heuristic search algorithm capable of generating large CDs on any number of alternatives, which also has the potential to discover unknown larger CDs. This is evidenced by the fact that the large CDs it found on the 10 and 11 alternatives improved the existing lower bounds. The success of our algorithm lies chiefly in its utilisation of human domain knowledge on this subject which existing

Table 10 The restriction sizes and their count for the 11 alternative CD of size 2349 when restricted to 4 to 10 alternative subsets

Subset n	Restriction sizes and their count									
4	Size	8	9							
	Count	143	187							
5	Size	16	17	18	19	20				
	Count	28	46	5	279	104				
6	Size	32	35	36	39	40	41	42	43	44
	Count	3	10	9	35	2	73	164	79	35
7	Size	74	79	85	86	87	88	89	91	92
	Count	3	3	12	3	9	3	15	19	44
8	Size	96	97	98	100					
	Count	39	52	42	19					
9	Size	179	184	187	192	194	196	200	201	202
	Count	3	1	1	1	3	8	3	2	8
10	Size	209	210	211	212	213	214	215	216	217
	Count	10	3	12	17	11	2	8	13	2
10	Size	415	426	431	448	451	452	457	460	466
	Count	1	1	1	2	2	1	1	1	4
10	Size	475	478	479	480	481	484	485		
	Count	3	4	3	4	11	4	4		
10	Size	1021	1026	1035	1045	1053	1068	1074	1078	
	Count	1	1	1	2	1	1	2	2	

The bold numbers indicate that they were previously unknown

learning algorithms for general combinatorial optimisation problems have no apparent way to assimilate. Our results were obtained without applying any statistic planning approaches, like Monte Carlo Tree Search, but by traversing the tree using a purely manually designed heuristic function to guide the search. The way we traversed the search tree only used a pure heuristic function but did not explicitly use any reinforcement learning techniques like Q-learning (Watkins and Dayan 1992). Despite having access to limited computational resources, our algorithm found many new large CDs.

Although the application of the database does cut away a large part of the search tree at a cheap cost, it has certain limitations. With a reasonable amount of computations, the algorithm could not find a new lower bound for CDs on 12 alternatives, suggesting that the CDs with a larger number of alternatives have more non-local implications, and the upper bound given by the database will more often differ from the actual size of the local restriction.

Finding the largest CDs is a challenging yet unsolved problem. Many existing algorithms can be applied to finding large CDs. We evaluated various standard approaches, including reinforcement learning algorithms, evolutionary algorithms and local search algorithms. In our preliminary experiments, they only managed to find local maxima solutions and had great difficulties finding the largest known domains for low values of n , such as $n = 7$ and $n = 8$. These algorithms did not benefit from longer execution times, as they were often trapped in local optima that are ubiquitous within the search space.

The new CDs we report suggest that there may be more complex mathematical patterns underlying the CDs than previously thought. Our results indicate that the maximal size of CDs on $n \geq 8$ alternatives can serve as a benchmark for heuristic search algorithms. The problems are challenging because the search space is enormous on large alternatives, and there are no obvious, simple criteria for searching. At the same time, many inherent structures in the CD can be exploited as the restrictions to each subset of alternatives also form a CD. An efficient search algorithm should be able to leverage this rich underlying structure to guide the search.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s10288-025-00583-1>.

Acknowledgements The authors would like to thank Klas Markström for his constructive comments on the manuscript. Bei Zhou was funded by the China Scholarship Council (CSC). This research utilised Queen Mary's Apocrita HPC facility (King et al. 2021), supported by QMUL Research-IT.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abello JM, Johnson CR (1984) How large are transitive simple majority domains? *SIAM J Algebr Discrete Methods* 5(4):603–618
- Akello-Egwel D, Leedham-Green C, Litterick A, Markström K, Riis S (2024) Condorcet domains on at most seven alternatives. *Math Soc Sci*
- Arrow KJ (1963) *Social choice and individual values*, vol 12, 2nd edn. Yale University Press
- Barrett T, Clements W, Foerster J, Lvovsky A (2020) Exploratory combinatorial optimization with reinforcement learning. In: *Proceedings of the AAAI conference on artificial intelligence*, vol 34, pp 3243–3250
- Bellman R (1966) Dynamic programming. *Science* 153(3731):34–37
- Danilov VI, Karzanov AV, Koshevoy G (2012) Condorcet domains of tiling type. *Discrete Appl Math* 160(7–8):933–940
- Danilov VI, Koshevoy GA (2013) Maximal Condorcet domains. *Order* 30:181–194
- Dittrich T (2018) *Eine vollständige klassifikation von condorcet domains für kleine alternativenmengen* (Unpublished doctoral dissertation). KIT-Bibliothek
- El-Shorbagy MA, Hassanien AE (2018) Particle swarm optimization from theory to applications. *Int J Rough Sets Data Anal (IJRSDA)* 5(2):1–24
- Fawzi A, Balog M, Huang A, Hubert T, Romera-Paredes B, Barekatin M et al (2022) Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* 610(7930):47–53
- Fishburn P (1996) Acyclic sets of linear orders. *Soc Choice Welfare* 14(1):113–124
- Fishburn PC (2002) Acyclic sets of linear orders: a progress report. *Soc Choice Welfare* 19:431–447
- Galambos A, Reiner V (2008) Acyclic sets of linear orders via the Bruhat orders. *Soc Choice Welfare* 30(2):245–264
- Karpov A (2022) Structured preferences: a literature survey. *Autom Remote Control* 83(9):1329–1354
- Karpov A, Slinko A (2022) Symmetric maximal Condorcet domains. *Order* 40:289–309
- Karpov A, Slinko A (2023) Constructing large peak-pit Condorcet domains. *Theor Decis* 94(1):97–120
- Karpov A, Markström K, Riis S, Zhou B (2023) Set-alternating schemes: a new class of large Condorcet domains. *arXiv preprint [arXiv:2308.02817](https://arxiv.org/abs/2308.02817)*
- Karpov A, Markström K, Riis S, Zhou B (2024) Local diversity of Condorcet domains. *arXiv preprint [arXiv:2401.11912](https://arxiv.org/abs/2401.11912)*
- King T, Butcher S, Zalewski L (2021) Apocrita—high performance computing cluster for queen Mary university of London (2017). Accessed 5 Feb
- Lackner M-L, Lackner M (2017) On the likelihood of single-peaked preferences. *Soc Choice Welfare* 48:717–745
- Laterre A, Fu Y, Jabri MK, Cohen A-S, Kas D, Hajjar K, Beguir K (2018) Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. *arXiv preprint [arXiv:1807.01672](https://arxiv.org/abs/1807.01672)*
- Lauri J, Dutta S, Grassia M, Ajwani D (2023) Learning fine-grained search space pruning and heuristics for combinatorial optimization. *J Heuristics* 29:313–347
- Leedham-Green C, Markström K, Riis S (2024) The largest Condorcet domain on 8 alternatives. *Soc Choice Welfare* 62(1):109–116
- Mankowitz DJ, Michi A, Zhernov A, Gelmi M, Selvi M, Paduraru C et al (2023) Faster sorting algorithms discovered using deep reinforcement learning. *Nature* 618(7964):257–263
- Markström K, Riis S, Zhou B (2024) Arrow's single peaked domains, richness, and domains for plurality and the Borda count. *arXiv preprint [arXiv:2401.12547](https://arxiv.org/abs/2401.12547)*
- Mazyavkina N, Sviridov S, Ivanov S, Burnaev E (2021) Reinforcement learning for combinatorial optimization: a survey. *Comput Oper Res* 134:105400
- Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG et al (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533
- Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: *International conference on machine learning*, pp 1928–1937
- Monjardet B (2009) Acyclic domains of linear orders: a survey. In: *The mathematics of preference, choice and order: essays in honor of Peter C. Fishburn*, pp 139–160
- Puppe C, Slinko A (2024) Maximal Condorcet domains a further progress report. *Games Econ Behav* 145:426–450
- Radhakrishnan A, Jeyakumar G (2021) Evolutionary algorithm for solving combinatorial optimization—a review. In: *Innovations in computer science and engineering: proceedings of 8th ICICSE*, pp 539–545

- Raynaud H (1981) Paradoxical results from Inada's conditions for majority rule. Institute for Mathematical Studies in the Social Sciences, Stanford University
- Raynaud H (1982) The individual freedom allowed by the value restriction condition. Institute for Mathematical Studies in the Social Sciences
- Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347)
- Sen AK (1966) A Possibility theorem on majority decisions. *Econometrica* 34(2):491–499
- Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A et al (2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362(6419):1140–1144
- Slinko A (2019) Condorcet domains satisfying arrow's single-peakedness. *J Math Econ* 84:166–175
- Slinko A (2024) A family of condorcet domains that are single-peaked on a circle. *Soc Choice Welfare* 1–11
- Watkins CJ, Dayan P (1992) Q-learning. *Mach Learn* 8:279–292
- Zhou B, Riis S (2022) Impartial games: a challenge for reinforcement learning. arXiv preprint [arXiv:2205.12787](https://arxiv.org/abs/2205.12787)
- Zhou B, Markström K, Riis S (2024) CDL: a fast and flexible library for the study of permutation sets with structural restrictions. *SoftwareX* 28:101951

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.