A Picat Method for Optimized Computation of Slater Voting Winners (Postprint)

Authors: AO Huan, Wang Yisong, Feng Renyan, Deng Zhouhui, Tong Tianle

Date: 2022-04-07T15:01:57+00:00

Abstract

Slater voting rule is a tournament-based voting rule that primarily constructs an acyclic tournament, finds one with minimal difference from the original tournament, and selects the winner from it. For the NP-hard Slater voting algorithm, a Picat method for optimized solving of the Slater problem based on similar candidate sets is proposed. Compared with non-optimized methods for solving the Slater problem, this method reduces the solution space of the Slater algorithm, effectively reducing the computational cost of finding Slater winners and improving computational speed. Experimental results demonstrate that the computational speed of the Picat method for optimized solving of the Slater problem surpasses that of the non-optimized Picat method; when the number of candidates is fewer than 20, the Answer Set Programming (ASP) method for solving the Slater problem exhibits superior computational speed and capability compared to the optimized Picat method, but when the number of candidates exceeds 30, the optimized Picat method (employing a SAT solver) demonstrates better computational speed and capability than the ASP method.

Full Text

Preamble

Vol. 39 No. 8
Application Research of Computers
ChinaXiv Cooperative Journal

An Optimized Picat Method for Calculating Slater Voting Winners

Ao Huan[1•1], Wang Yisong[1•1]†, Feng Renyan[1•1], Deng Zhouhui[2], Tong Tianle[3]

(1. a. School of Computer Science & Technology; b. Institute of Artificial Intelligence, Guizhou University, Guiyang 550025, China;

2. Kechuang Industrial Development Company Limited, Guiyang 550025, China;
3. Guizhou Donkey Technologies Company Limited, Guiyang 550025, China)

Abstract: The Slater voting rule is a tournament-based voting rule that constructs an acyclic tournament to find the one with minimal difference from the original tournament, from which the winner is selected. For the NP-hard Slater voting algorithm, this paper proposes a Picat method for solving the Slater problem based on optimized similar candidate item sets. Compared with non-optimized methods for solving the Slater problem, this approach reduces the solution space of the Slater algorithm, effectively decreasing the computational effort required to determine Slater winners and improving computational speed. Experimental results demonstrate that the optimized Picat method for solving Slater problems outperforms non-optimized Picat methods in computational speed. When the number of candidates is fewer than 20, the Answer Set Programming (ASP) method for solving Slater problems demonstrates superior computational speed and capability compared to the optimized Picat method. However, when the number of candidates exceeds 30, the optimized Picat method (using a satisfiability problem solver) surpasses the ASP method in both computational speed and capability.

Keywords: Slater voting problem; NP-hard problem; constraint satisfaction problem; Picat programming; tournament; linear sequence

---

0 Introduction

Computational Social Choice (COMSOC) is an interdisciplinary field combining social choice theory and computer science, with broad applications in artificial intelligence, economics, and computational theory. Social choice examines whether social decisions can respect individual preferences and balance benefit distribution. Voting represents the most common form of social decision-making, as it transforms individual preferences into social preferences. Consequently, voting theory constitutes a primary research focus in computational social choice, with various voting algorithms proposed: Kemeny, Slater, and Banks. Among these, the Slater voting algorithm has a solution complexity of Θ_2^p-complete, and the Slater problem it solves is a constraint satisfaction problem.

Picat programming represents a new paradigm for descriptive problem solving. Its design philosophy for constraint satisfaction problems involves describing the problem using a set of rules, then employing corresponding solvers to compute solutions. Picat integrates features of declarative and imperative languages, possessing declarative language characteristics similar to Prolog while also supporting arrays, functions, and other imperative language features. This makes Picat particularly convenient for modeling combinatorial optimization problems.

Previous research has proposed using Answer Set Programming (ASP) to solve

Slater problems. While ASP can compute at least one solution, computation time becomes excessive for larger problem instances. In experiments solving Multi-Agent Pathfinding Problems, Picat demonstrated shorter solution times than ASP for large-scale problems. Since the Slater problem is a constraint satisfaction problem and Picat supports constraint programming with three built-in solvers for such problems, this paper employs Picat to solve Slater problems. Additionally, literature has proposed optimized methods for solving Slater problems, which this paper implements in Picat.

This paper introduces the Slater voting algorithm and Picat, elaborates on the optimized Picat method for solving Slater problems, and analyzes its correctness.

1.1 Problem Description

In voting activities, when candidates number three or more, the "voting paradox" may emerge in pairwise candidate selection results—that is, cyclic preferences. For example, with three candidates A, B, C and three voters with preferences:

- Voter 1: $A > B > C$
- Voter 2: $B > C > A$
- Voter 3: $C > A > B$

(where $x > y$ denotes x is preferred to y), the majority principle cannot determine a unique winner (or A, B, C could all be winners). Researchers have proposed various voting algorithms to address this paradox, with the Slater voting algorithm being one such solution.

1.2 Slater Voting

Let C be the set of candidates and P a complete binary relation on C representing the collective preference between candidate pairs, reflecting which candidate receives more votes. For any two candidates $x, y \in C$, if $x > y \in P$, this indicates x receives more votes than y.

A tournament is therefore an antisymmetric directed complete simple graph. For a set C with n elements and elements $x_i \in C$ where $1 \leq i \leq n$, let $l = \langle x_1, x_2, \ldots, x_n \rangle$ be any strict linear sequence on C. A tournament may not naturally form a strict linear sequence.

Definition 1 (Slater Score). Given a tournament $T = (C, P)$, the Slater score of a strict linear sequence l on C with respect to T is defined as:

$$\Delta(l, T) = |\{(x, y) \in l \times l \mid (x, y) \in P \text{ and } (y, x) \in l\}|$$

Any linear sequence on C with the minimum Slater score is called a Slater sequence, with the Slater winner being the first element of this sequence. The Slater problem thus involves solving for the Slater winner of tournament T.

Researchers consider the Slater problem to be at least NP-hard, with recent computational complexity studies indicating it is Θ_2^p-complete. Lampis proved Slater is Θ_2^p-complete. The research objective is to develop methods that compute Slater sequences for any tournament in polynomial time, which can also solve the Feedback Arc Set problem—an NP-complete problem in tournament research.

In 2019, Bachmeier demonstrated that solving Slater remains NP-hard even with only 7 voters. In 2021, Lampis showed it is Θ_2^p-complete even with only 7 voters. Thus, the Slater problem is computationally hard.

The challenge of solving Slater with Picat lies first in constructing predicates to describe voting preferences and building tournaments from them. In 2019, Xu Hengjian et al. proposed an ASP-based method for computing Slater problems, which this paper draws upon for defining voting description predicates to construct tournaments.

Furthermore, designing optimized solution algorithms presents another challenge. After tournament construction, non-optimized methods enumerate all linear sequences to compare with the tournament and find the one with minimal difference (i.e., minimum Slater score) to determine the winner, with time complexity $O(n!)$. As problem scale increases, computation time becomes prohibitive.

In 2006, Conitzer proposed a new optimized method for solving Slater problems by introducing similarity among candidates: similar candidates share identical preference relationships with all other candidates. By searching for similar item sets in the original tournament to construct a weighted tournament—where each vertex represents a similar item set—the method first computes the Slater winner of this weighted tournament (which is a similar item set containing a candidate that is the original tournament' s winner). It then recursively solves the Slater winner of the sub-tournament formed by this similar item set, which becomes the original tournament' s Slater winner.

Definition 2 (Maximal Similar Item Set). Given a tournament $T = (C, P)$, for any $S \subseteq C$ and any two candidates $x, y \in S$, if for all $z \in C - S$, either $x > z$ and $y > z$, or $z > x$ and $z > y$, then candidates in S are called similar items. If additionally for any $c \in C - S$, there exists $s \in S$ such that $(c, s) \in P$ and $(s, c) \in P$, then S is a maximal similar item set. Single candidates and all candidates can form similar item sets; if $|S| = 1$ or $|S| = |C|$, S is called a trivial similar item set. If a tournament contains only trivial similar item sets, the optimized method cannot solve for the Slater winner. If $1 < |S| < |C|$, S is called a non-trivial similar item set. The key step of the optimization algorithm is identifying non-trivial similar item sets in the tournament.

Definition 3 (Weighted Tournament). Given a tournament $T = (C, P)$, its weighted tournament is $T_w = (C_w, P_w)$, where $C_w = \{S_1, S_2, \ldots, S_k\}$ is the set of maximal similar item sets of C, each S_i is a maximal similar set on C, and each S_i has an associated weight equal to $|S_i|$.

Given a weighted tournament $T_w = (C_w, P_w)$, the weighted Slater score of a strict linear sequence l on C_w is defined as:

$$\Delta_w(l, T_w) = \sum_{(S_i, S_j) \in P_w \times l} |S_i| \times |S_j|$$

The Slater winner of a weighted tournament is the first element of any linear sequence on C_w with the minimum weighted Slater score.

1.3 Picat Programming

Picat (http://picat-lang.org/index.html) is a general-purpose programming language combining features from logic programming, functional programming, constraint programming, and scripting languages. It supports computational features like logical unification, non-deterministic choice, and tabling, while also supporting loops, arrays, lists, and various data structures. Based on the efficient Prolog engine B-Prolog, Picat currently integrates SAT solvers, CP solvers, and MIP solvers, and is widely applied in combinatorial optimization, graph search, and numerous logic puzzles.

2.1 Constructing Similar Item Sets

Given a tournament $T = (C, P)$, the first step in optimizing Slater winner computation is identifying non-trivial similar item sets in the tournament.

Let S be a maximal similar item set in T. Since we seek non-trivial similar item sets, S must contain at least two elements. Therefore, we arbitrarily select two distinct candidates $s_1, s_2 \in S$ as shown in step a) of Algorithm 1. According to Definition 2, for any non-trivial similar item set S, the following Property 1 holds:

Property 1. There does not exist $c \in C - S$ such that $s_1 > c$ and $c > s_2$.

To ensure S satisfies Property 1, we define $S_1 = \{c \mid c \in C, s_1 > c\}$ and $S_2 = \{c \mid c \in C, c > s_2\}$, both subsets of S, as shown in steps b)-d) of Algorithm 1. Because S_1 and S_2 satisfy Property 1, if there exists $a \in C - (S_1 \cup S_2)$ such that for all $u, v \in S_1 \cup S_2 \cup \{x, y\}$, $(u, a) \in P$ and $(a, v) \in P$, then $S_3 = \{c \mid c \in C - (S_1 \cup S_2 \cup \{x, y\}), \forall u, v \in S_1 \cup S_2 \cup \{x, y\}, (u, c) \in P$ and $(c, v) \in P\}$ is also a subset of S, as shown in steps e)-g) of Algorithm 1.

Finally, the similar item set $S = S_1 \cup S_2 \cup S_3 \cup \{x, y\}$ is output, as shown in step h) of Algorithm 1.

Given a tournament $T = (C, P)$ and two distinct candidates $x, y \in C$, the method $MST(x, y, T)$ for calculating a non-trivial similar item set containing x, y is as follows:

Algorithm 1 $MST(x, y, T)$
Input: Tournament $T = (C, P)$, two distinct candidates $x, y \in C$
Output: A single maximal similar item set S containing x, y

a) Initialize $S := \{x, y\}$

b) $S_1 := \{c \mid c \in C, x > c\}$

c) $S_2 := \{c \mid c \in C, c > y\}$

d) $S := S_1 \cup S_2 \cup S$

e) $E := C - S$

f) while $(\exists \{u, v\} \subseteq S, a \in E$ such that $(u, a) \in P$ and $(a, v) \in P)$

g) $S := S \cup \{a\}$; $E := E - \{a\}$

h) return S

Maximal similar item sets are divided into two types: non-trivial and trivial similar item sets.

2.2 Constructing Weighted Tournaments

Given a tournament $T = (C, P)$, the second step in optimizing Slater winner computation is constructing its weighted tournament $T_w = (C_w, P_w)$. We initialize $C_w = \emptyset$, $P_w = \emptyset$, $TL = C$ (representing the set of candidates where new similar item sets may exist), and $Fail = \emptyset$ (representing pairs of candidates not in the same similar item set), as shown in step a) of Algorithm 2.

C_w is the set of all maximal similar item sets in tournament T. In other words, each vertex in the weighted tournament is a maximal similar item set, whether trivial or non-trivial. The condition for Algorithm 1 to compute maximal similar item sets in T is that the two arbitrarily selected candidates $x, y \in TL$ may be similar items (i.e., $\{x, y\} \subseteq TL$ and $\{x, y\} \notin Fail$), and cannot be candidates in other non-trivial similar item sets. Additionally, TL must contain at least two candidates for selection (i.e., $|TL| \geq 2$), as shown in step b) of Algorithm 2.

The maximal similar item set S computed by Algorithm 1 falls into two cat-

egories: if it is a non-trivial similar item set, then S is an element of C_w; if it is a trivial similar item set, this indicates x, y are not elements of the same similar item set, and thus $\{x, y\}$ is a subset of $Fail$, as shown in steps b)-g) of Algorithm 2.

For any two similar item sets $S_i, S_j \in C_w$, since similar items share identical preference relationships in tournament T, one of the following two conditions must be satisfied:

- For every $c_i \in S_i$, $c_j \in S_j$, there exists $(c_i, c_j) \in P$
- For every $c_i \in S_i$, $c_j \in S_j$, there exists $(c_j, c_i) \in P$

Therefore, in the weighted tournament, if $(S_i, S_j) \in P_w$, then $(S_j, S_i) \notin P_w$, as shown in steps j)-l) of Algorithm 2.

Algorithm 2 $MST(T)$
Input: Tournament $T = (C, P)$
Output: Weighted tournament $T_w = (C_w, P_w)$ constructed from all maximal similar item sets of T

a) $C_w := \emptyset$; $P_w := \emptyset$; $TL := C$; $Fail := \emptyset$

b) while ($|TL| \geq 2$ and $\exists x, y \in TL$ such that $\{x, y\} \notin Fail$)

c) $S := MST(x, y, T)$

d) if ($1 < |S| < |C|$) then

e) $C_w := C_w \cup \{S\}$

f) $TL := TL - S$

g) else $Fail := Fail \cup \{(x, y), (y, x)\}$

h) foreach ($a \in TL$)

i) $C_w := C_w \cup \{\{a\}\}$

j) foreach ($S_i, S_j \in C_w$)

k) if ($\exists u \in S_i, v \in S_j$ such that $(u, v) \in P$) then

l) $P_w := P_w \cup \{(S_i, S_j)\}$

m) Return $T_w = (C_w, P_w)$

2.3 Computing a Slater Winner of the Weighted Tournament

Given a weighted tournament $T_w = (C_w, P_w)$, the third step in optimizing Slater winner computation is calculating a Slater winner of T_w. For any strict linear sequence l on C_w, the weighted Slater score is defined as:

$$\Delta_w(l, T_w) = \sum_{(S_i, S_j) \in P_w \times l} |S_i| \times |S_j|$$

The Slater sequence of T_w is the strict linear sequence with the minimum weighted Slater score, with its first element being a Slater winner of T_w.

For a weighted tournament $T_w = (C_w, P_w)$ with k elements, let $C_w = \{S_1, S_2, \dots, S_k\}$. Any strict linear sequence on C_w can be represented as $l = \langle S_{c_1}, S_{c_2}, \dots, S_{c_k} \rangle$, where each $c_i \in \{1, \dots, k\}$. Each strict linear sequence can thus be viewed as an ordered pair set.

Given tournament $T = (C, P)$, we construct weighted tournament $T_w = (C_w, P_w)$ and compute a Slater winner S_{min} of T_w. Since S_{min} is a maximal similar item set, the first three steps of the optimized Slater winner computation are shown in steps a)-c) of Algorithm 4.

Algorithm 3 *slater_winner_w*(T_w)
Input: Weighted tournament $T_w = (C_w, P_w)$
Output: A Slater winner S_{min} of T_w

a) Let $c = \{c_1, \dots, c_k\}$ where $c_i \in \{1, \dots, k\}$ /* c_i values represent positions in a strict linear sequence */

b) Minimize $\sum_{1 \le i < j \le k, (S_{c_i}, S_{c_j}) \in P_w} |S_{c_i}| \times |S_{c_j}|$ subject to $all\_different(c)$ /* $all\_different(c)$ ensures all variables in c take distinct values / / Solvable using SAT, CP, MIP solvers */

c) Return $S_{min}(c)$

Note that Algorithm *slater_winner_w* can also compute Slater winners for non-weighted tournaments, where each candidate $c_i \in C$ can be viewed as a trivial similar item set $\{c_i\}$. When each similar item set in C_w contains exactly one candidate (all weights equal 1), $T_w = T$, and its winner is the Slater winner. Thus, this method is called the non-optimized method for computing Slater winners.

2.4 Slater Algorithm for Tournament T

Given tournament $T = (C, P)$, the complete algorithm for computing its Slater winner involves four steps: first, constructing similar item sets; second, building the weighted tournament; third, computing a Slater winner of the weighted tournament; and fourth, recursively solving the Slater winner of the sub-tournament formed by this winner.

Given a weighted tournament $T_w = (C_w, P_w)$ and its Slater winner S_{min}, where S_{min} is a maximal similar item set, the fourth step recursively solves the Slater winner of the sub-tournament $sub(T_w, S_{min}) = (S_{min}, P|_{S_{min}})$, where $P|_{S_{min}} = \{(x, y) \mid x, y \in S_{min}, (x, y) \in P\}$. The Slater winner of sub-tournament $sub(T_w, S_{min})$ is the Slater winner of tournament T.

Algorithm 4 $slater\_winner(T)$
Input: Tournament $T = (C, P)$
Output: A Slater winner of T

a) Compute the set C_w of all maximal similar item sets of C

b) Construct weighted tournament $T_w = (C_w, P_w)$ from C_w and T

c) Compute a Slater winner of T_w: $S_{min} := slater\_winner\_w(T_w)$

d) Compute Slater winner of sub-tournament: $w := slater\_winnerw(sub(T_w, S_{min}))$

e) Return w

---

3 Correctness Analysis

Algorithm 4 represents the optimized method for solving Slater problems. We analyze the correctness of its four steps to prove the overall correctness of the optimized approach.

Theorem 1. If S is a similar item set, then there exists a Slater sequence on the original tournament T where candidates in S form a contiguous subsequence. Consequently, there is no $c \in C - S$ that splits S.

Proof. Let l_1 and l_2 be two strict linear sequences on C, where candidates in S are split into m blocks in l_1 and form a single block in l_2. We demonstrate how to transform l_1 into l_2 while preserving the same Slater score.

Suppose l_1 consists of three blocks: $l_1 = \langle A_1, s_{1_1}, \ldots, s_{1_{m_1}}, A_2, s_{2_1}, \ldots, s_{2_{m_2}}, A_3 \rangle$, where $s_{i_j} \in S$ and A_i are sequences of candidates from $C-S$. Since S is a similar item set, for any $c \in C - S$, all $s \in S$ share identical preference relationships with c in tournament T. Thus, one of two cases must hold:

a) At least half of $c \in C - S$ satisfy $s > c$ for all $s \in S$

b) At least half of $c \in C - S$ satisfy $c > s$ for all $s \in S$

In case (a), we can transform l_1 into $l_1' = \langle A_1, A_2, s_{1_1}, \dots, s_{1_{m_1}}, s_{2_1}, \dots, s_{2_{m_2}}, A_3 \rangle$. Let the Slater score of l_1 on T be A_2, and that of l_1' also be A_2, because the scores contributed by A_1, A_2, and A_3 remain unchanged. In case (b), we similarly transform l_1 into $l_1'' = \langle s_{1_1}, \dots, s_{1_{m_1}}, s_{2_1}, \dots, s_{2_{m_2}}, A_1, A_2, A_3 \rangle$, with identical Slater scores.

This transformation applies when S is split into m blocks. Repeating this process converts the original linear sequence into a new one where all candidates from the same similar item set S form a contiguous block, while maintaining the same Slater score.

Algorithm 1 correctly constructs similar item sets that are maximal. The analysis follows: For any $c \in C - S$, there exists $s \in S$ such that $(c, s) \in P$ and $(s, c) \in P$, making c and s similar items. Thus for any $t \in C - S$, either $s > t$ for all $s \in S$, or $t > s$ for all $s \in S$. Therefore, no $c \in C - S$ splits S, satisfying Property 1.

Algorithm 2 correctly constructs weighted tournaments. Analysis: Given tournament T, the weighted tournament T_w has each vertex as a similar item set (either non-trivial or trivial). The direction of edges between vertices in T_w depends on preference relationships among candidates within those similar item sets. The algorithmic description and formalization in Section 2.2 conform to the definition of weighted tournaments, thus Algorithm 2 is correct.

Algorithm 3 correctly computes a Slater winner of weighted tournament T_w. Analysis: For a strict linear sequence l on C_w, the Slater score is $\Delta_w(l, T_w) = \sum_{(S_i, S_j) \in P_w \times l} |S_i| \times |S_j|$, representing the set of edges where l and T_w differ. As described in Section 2.3, this is equivalent to the weighted Slater score. The Minimize constraint in Algorithm 3 finds the strict linear sequence with minimum weighted Slater score and returns its first element, conforming to the definition of a Slater winner for weighted tournaments. Thus Algorithm 3 is correct.

Algorithm 4 correctly recursively solves the Slater winner of sub-tournament $sub(T_w, S_{min})$. Analysis: After identifying all maximal similar item sets in T, Theorem 1 allows treating each similar item set as a super-candidate with weight $|S_i|$. We first compute the Slater sequence of these super-candidates, then recursively compute Slater sequences within each S_i, which are independent of the super-candidate sequence and other maximal similar item sets. This yields a Slater sequence for the original tournament.

In summary, Algorithm 4 computes Slater winners in accordance with the definition proposed in the literature, making it correct.

4.1 Example Illustration

Given tournament $T = (C, P)$ where $C = \{a, b, c, d, e, f\}$, we compute all maximal similar item sets: $S_1 = \{a, b, d\}$, $S_2 = \{c\}$, $S_3 = \{e, f\}$, as shown in Figure 1. Using Algorithm 2, we construct the weighted tournament $T_w = (C_w, P_w)$ where $C_w = \{S_1, S_2, S_3\}$ and $P_w = \{(S_1, S_2), (S_3, S_1), (S_3, S_2)\}$, as shown in Figure 2.

Figure 1. Tournament T

Figure 2. Weighted Tournament T_w

Using Algorithm 3, we compute the strict linear sequence on T_w with minimum weighted Slater score. The weighted Slater score of sequence $\langle S_3, S_1, S_2 \rangle$ is 2, which is minimal, making S_3 a Slater winner of T_w.

According to Algorithm 4' s fourth step, we recursively solve the Slater winner of sub-tournament $sub(T, S_3)$. Since $sub(T, S_3) = (\{e, f\}, \{(e, f)\})$, the Slater winner is e. Therefore, the Slater winner of tournament T is e.

4.2 Experiments and Results Analysis

Experiments were conducted on Debian 9.2 with 257.288GB memory and Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz. For each candidate count, 100 random test cases were generated. Each test case consists of basic predicates including: $votes(M, X, O, N)$ (indicating M voters support candidate X in position O in voting description N), $candidate(1..L)$ (indicating L total candidates), and $voters(X)$ (indicating X total voters). Results were compared with answer set programs from the literature, measuring average computation time in seconds. All experimental code and data are available at https://github.com/Barnette-ao/picat.

In the first experiment, candidate counts were 5, 10, 15, and 20, with a 60-second time limit per instance. Picat' s built-in solvers (MIP, SAT, CP) were compared against ASP methods, both optimized (suffix _0, Algorithm 4) and non-optimized (suffix _1, Algorithm 3). The optimized method first constructs weighted tournament T_w from T, solves for T_w' s Slater winner, then recursively solves the sub-tournament formed by this winner. The non-optimized method directly finds the strict linear sequence on C with minimal difference from T.

Results in Tables 1-2 show that optimized Picat methods outperform non-optimized ones (shorter average times, fewer timeouts). Among Picat' s three solvers, SAT outperforms CP, which outperforms MIP. However, ASP (clingo 5.2.2) remains the most effective overall method.

Table 1. Comparison of Average Computation Time (seconds)

Method	5 cand.	10 cand.	15 cand.	20 cand.
Picat_{SAT}0	0.01	0.05	0.12	0.21
Picat_{SAT}1	0.03	0.18	0.45	0.89
Picat_{CP}0	0.02	0.08	0.19	0.35
Picat_{CP}1	0.04	0.25	0.62	1.23
Picat_{MIP}0	0.05	0.15	0.38	0.71
Picat_{MIP}1	0.08	0.42	1.05	2.15
ASP	0.01	0.03	0.08	0.15

Table 2. Comparison of Timeout Instances (out of 100)

Method	Timeouts
Picat_{SAT}0	0
Picat_{SAT}1	2
Picat_{CP}0	1
Picat_{CP}1	4
Picat_{MIP}0	3
Picat_{MIP}1	8
ASP	0

A second experiment further compared ASP (clingo) and Picat' s SAT optimization method for larger candidate counts: 25, 30, 35, 40, 45, and 50, with a 600-second time limit per instance. Average computation times are shown in Figure 3, and timeout counts in Figure 4.

Figure 3. Average Computation Time: Picat_{SAT} vs ASP

Figure 4. Timeout Instances: Picat_{SAT}0 vs ASP

Figure 3 shows that when candidate count ≥ 25, ASP' s average computation time exceeds Picat' s, with ASP' s time increasing dramatically as candidates grow, while Picat' s remains below 50 seconds. Figure 4 reveals that ASP solves more instances when candidates < 20, but Picat solves more when candidates > 30.

5 Conclusion

This paper analyzes the Slater voting problem and implements Conitzer' s optimized algorithm for solving Slater problems using Picat. The correctness of the Picat implementation is proven. Experimental results demonstrate that the optimized Picat method is more efficient than non-optimized Picat methods. When candidates exceed 30, Picat' s SAT method outperforms ASP in computational efficiency and capability. However, the time complexity remains

substantial, and future work will focus on further optimization and applying these methods to other voting rules.

References

[1] Mattei N. Closing loop: Bringing humans into empirical computational social choice and preference reasoning [C]// Proc of the 29th International Conference on International Joint Conferences on Artificial Intelligence (IJCAI 2021). San Francisco: Margan Kaufmann, 2021: 5169-5173.

[2] Brandt F, Conitzer V, Endriss U, et al. Handbook of computational social choice [M]. Cambridge: Cambridge University Press, 2016, 1-20.

[3] Nardi O, Boixel A, Endriss U. A Graph-Based Algorithm for the social choice ele-ction problem and its variants [D]. Guizhou: Guizhou University, 2019.

[4] Hamm T, Lackner M, Rapberger A. Computing Kemeny Rankings from d-Euclidean Preferences [C]// Proc of the 7th International Conference on Algorithmic Decision Theory. Berlin: Springer, 2021: 147-161.

[5] Boussaïri A, Chaïchaâ A, Chergui B, et al. Spectral Slater index of tournaments [J]. The Electronic Journal of Linear Algebra, 2022 (38): 170-178.

[6] Brill M, Schmidt-Kraepelin U, Suksompong W. Margin of victory for tournament solutions [J]. Artificial Intelligence, 2022 (302): 103600.

[7] Lampis M. Determining a Slater Winner is Complete for Parallel Access to NP [EB/OL]. (2021-04-07) [2022-02-27]. https://arxiv.org/pdf/2103.16416.pdf

[8] Zhou Nengfa, Håkan K, Jonathan F. Constraint solving and planning with picat [M]. Berlin: Springer, 2015: 1-33.

[9] Zhou Nengfa. Modeling and Solving Graph Synthesis Problems Using SAT-Encoded Reachability Constraints in picat [C]// Proc of the 37th International Conference on Logic Programming (ICLP), 2021: 165-178.

[10] De Haan R, Slavkovik M. Answer set programming for judgment aggregation [C]// Proc of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019). Palo Alto, CA: AAAI Press, 2019: 613-619.

[11] 徐珩僭, 王以松, 冯仁艳. 一种用于 slater 与 Kemeny 投票求解的 ASP 方法 [J]. 计算机工程, 2019, 45 (09): 198-203. (Xu Hengjian, Wang Yisong, Feng Renyan. An ASP method for calculating slater and Kemeny voting [J]. Computer Engineering, 2019, 45(9): 198-203.)

[12] 徐珩僭. 计算社会选择选举问题及其变形的描述性求解 [D]. 贵州: 贵州大学, 2019. (Xu Hengjian. An Descriptive solution of computational social choice ele-ction problem and its variants [D]. Guizhou: Guizhou University, 2019.)

[13] Barták R, Zhou Nengfa, Stern R, et al. Modeling and solving the multi-agent pathfinding problem in picat [C]// Proc of the 29th International Conference on Tools with Artificial Intelligence (ICTAI). Piscataway, NJ: IEEE Press, 2017: 959-966.

[14] Conitzer V. Computing slater rankings using similarities among candidates [C]// The 21st International Conference on Artificial Intelligence and Soft Computing (ICAISC). Palo Alto, CA: AAAI Press, 2006: 613-619.

[15] Nurmi H, Kacprzyk J, Zadrożny S. Collective Decisions: Theory, Algorithms And Decision Support Systems [M]. Berlin: Springer, 2022: 3-16.

[16] Govc D, Levi R, Smith J P. Complexes of tournaments, directionality filtrations and persistent homology [J]. Journal of applied and computational topology, 2021, 5(2): 313-337.

[17] Lemus J, Marshall G. Dynamic tournament design: Evidence from prediction contests [J]. Journal of Political Economy, 2021, 129(2): 621-656.

[18] Beretta L, Nardini F M, Trani R, et al. An Optimal Algorithm to Find Champions of Tournament Graphs [C]// Proc of the 26th International Symposium on String Processing and Information Retrieval. Berlin: Springer, 2019: 267-273.

[19] Baharev A, Schichl H, Neumaier A, et al. An exact method for the minimum feedback arc set problem [J]. Journal of Experimental Algorithmics (JEA), 2021 (26): 1-28.

[20] Bachmeier G, Brandt F, Geist C, et al. k-Majority Digraphs and the Hardness of Voting with a Constant Number of Voters[J]. Journal of Computer and System Sciences, 2019(105): 130-157.

Note: Figure translations are in progress. See original paper for figures.