S4. Queen Domination Problem

(Some Notes On William Herbert Bird's Thesis)

Taha Rostami

November, 2023

Some Clarifications

- The work is a good one, but it is easy to be misinterpreted. Thus, this work should be approached with caution; otherwise, it is easy to misunderstand what he actually did and what he actually intended to do.
- The Goal was to develop a specific solver for domination set problems. [not for other problems, not for any specific class of domination set problem (e.g., queen domination problem), etc.]
- The formulation was based on seeing the problem as an optimization problem (not a decision problem). [Thus, for example, their approach might not be best suited for domination set problems that we already know the size of minimum dom. set, but one configuration]

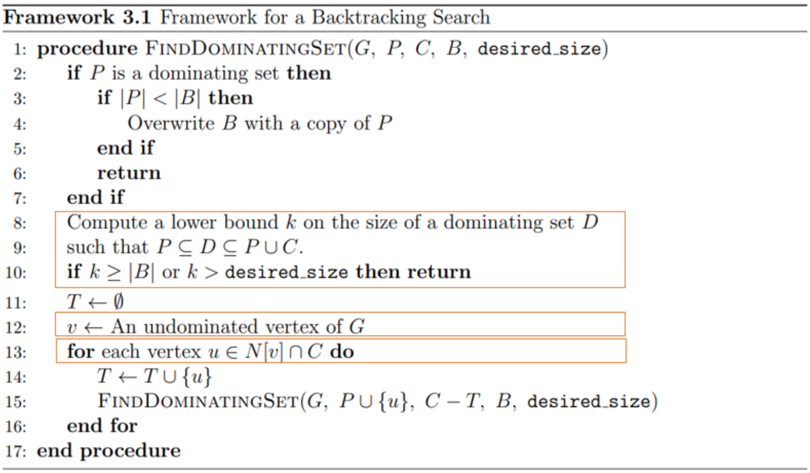
Some Clarifications

- In a large portion of the thesis, this work evaluates the proposed method on several classes of domination set problems. [Therefore, in any sense, Queen Dom. Problem was not the only interest of this work]
- However, after showing the general power of the proposed method, they dedicated some sections to solve open cases of a few particular classes of Dom. Set Problem, i.e., $\gamma(Q_n)$,$i(Q_n)$, and $b(Q_n)$. [The reason is unclear, but they might wanted to assess the ability of their method to solve open cases.]
- To solve open-cases of $\gamma(Q_n)$, and $i(Q_n)$ they literally used their proposed methods without any modification. However, To solve open cases of $b(Q_n)$, they proved some theorems and modified their methods considering some border rotation symmetries. [The reasons for their particular decisions are unclear]

Some Clarifications

- Regarding Queen Dom. and its first open case: "The smallest open case of the queen domination and independent domination problems is now n = 26. This case should be within the reach of the algorithm used for the open cases in this thesis with currently available research computing clusters (which, for funding reasons, were not leveraged for this research). Such clusters typically have thousands of processors, and all of the infrastructure for massive parallelization is already available in the implementations presented by this research (including the unidom program)."

"Framework 3.1 Framework for a Backtracking Search"



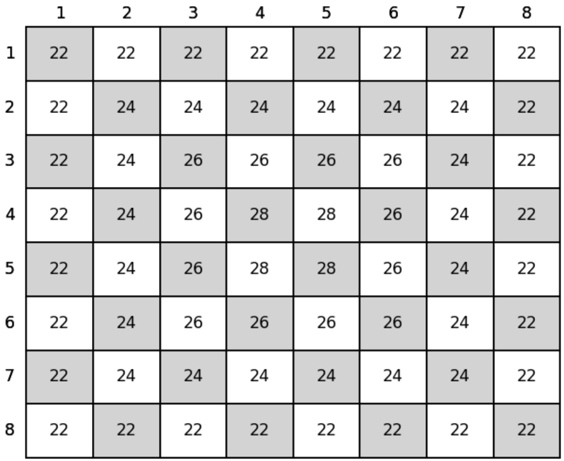
Framework 3.1 Framework for a Backtracking Search

```
procedure FindDominatingSet($G$, $P$, $C$, $B$, desired_size)
    if $P$ is a dominating set then
        if $|P| < |B|$ then
            Overwrite $B$ with a copy of $P$
        end if
        return
    end if
    Compute a lower bound $k$ on the size of a dominating set $D$
    such that $P \subseteq D \subseteq P \cup C$.
    if $k \geq |B|$ or $k >$ desired_size then return
    $T \leftarrow \emptyset$
    $v \leftarrow$ An undominated vertex of $G$
    for each vertex $u \in N[v] \cap C$ do
        $T \leftarrow T \cup \{u\}$
        FindDominatingSet($G$, $P \cup \{u\}$, $C - T$, $B$, desired_size)
    end for
end procedure
```

"Framework 3.1 Framework for a Backtracking Search" (Cont.)

- The problem formulation (opt. vs dec.) and targeting diff. classes can make a significant diff
- If optimization without a particular focus on specific classes like Queen Dom was intended, initialization of B using an approximation algorithm could probably significantly enhance the algorithm's running time in practice.
- their vertex ordering/selection heuristics are not optimized for queen dom. [for non-existence, it should not make much of a difference, but for existence, a key observation is that in optimal queen dom sol, queens are not placed in a row]

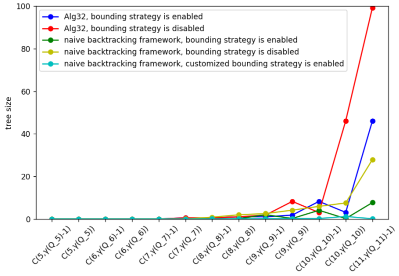
"Bounding Strategy 3.1." + A custom Bounding Strategy

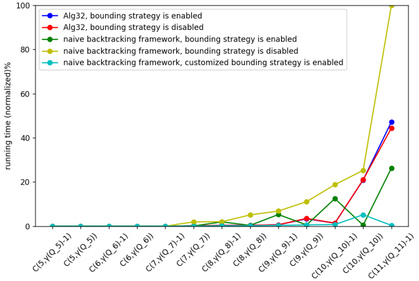


	1	2	3	4	5	6	7	8
1	22	22	22	22	22	22	22	22
2	22	24	24	24	24	24	24	22
3	22	24	26	26	26	26	24	22
4	22	24	26	28	28	26	24	22
5	22	24	26	28	28	26	24	22
6	22	24	26	26	26	26	24	22
7	22	24	24	24	24	24	24	22
8	22	22	22	22	22	22	22	22

Eval

- When comp. two algos, bigger/smaller tree size does not necessarily mean higher/lower running time.
- Below algorithms all are exponential/sub-exponential.
- The following evals. might not be valid for larger n

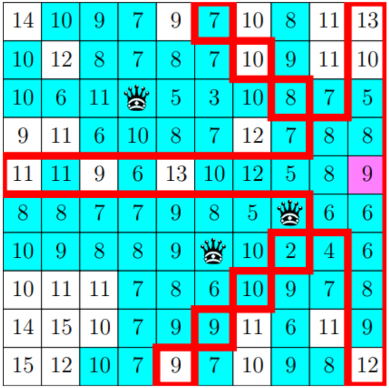




"Bounding Strategy 3.3."

5	10	♛	9	8	9	7	5
12	9	10	10	12	10	11	6
12	13	6	13	11	13	9	9
11	11	13	11	14	10	12	7
10	13	10	15	10	13	9	9
13	12	13	11	15	10	12	5
11	14	9	13	10	13	9	9
10	8	6	7	9	7	9	♛

"Bounding Strategy 3.6."



14	10	9	7	9	7	10	8	11	13
10	12	8	7	8	7	10	9	11	10
10	6	11	♛	5	3	10	8	7	5
9	11	6	10	8	7	12	7	8	8
11	11	9	6	13	10	12	5	8	9
8	8	7	7	9	8	5	♛	6	6
10	9	8	8	9	♛	10	2	4	6
10	11	11	7	8	6	10	9	7	8
14	15	10	7	9	9	11	6	11	9
15	12	10	7	9	7	10	9	8	12

Some Bugs

```
procedure FindDominatingSet($G$, $P$, $C$, $B$, desired_size)
    $n \leftarrow |V(G)|$
    if $|N[P]| = n$ then
        if $|P| < |B|$ then
            Overwrite $B$ with a copy of $P$
        end if
        return
    end if
    $k \leftarrow$ MinVerticesNeeded($G$, $P$, $C$)
    if $k \geq |B|$ or $k >$ desired_size then return
    $v \leftarrow$ ChooseNextVertex($G$, $P$, $C$)
    NeighbourList $\leftarrow N[v]$
    Sort NeighbourList by domination degree
    $F \leftarrow$ Empty stack
    for each vertex $v_j$ in NeighbourList do
        if $v_j \in C$ then
            Remove $v_j$ from $C$
            force_stop $\leftarrow$ FindDominatingSet($G$, $P \cup \{u\}$, $C$, $B$, desired_size)
            Push $j$ onto $F$
            if force_stop = true then
                Break
            end if
        end if
    end for
    while $F$ is non-empty do
        $j \leftarrow$ Pop(F)
        RestoreCandidate($G$, $P$, $C$, $B$, desired_size, $j$)
    end while
end procedure
```

non-trivial bug

Trivial bug

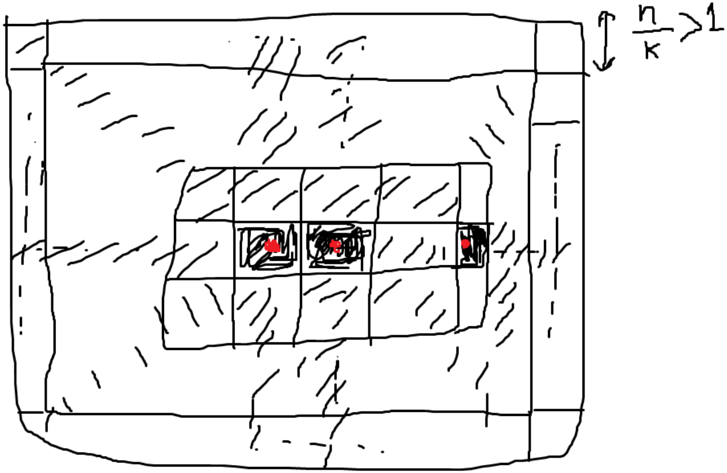
Conclusion and Potential Future Directions

- In Queen Dom. Problem, adding new constraints is a tricky task. This is because, in addition to the power of the constraints in terms of reducing the state space, they should also be lightweight for computation. Otherwise, their potential benefit can be outweighed by the required computational overhead.
- TODO: "Practical algorithms to compute automorphism groups (as a set of generators) are already widely used (notably, the nauty program [49]). Previous research by Myrvold and Fowler [50] and Bird and Myrvold [5] has presented practical approaches to isomorphism elimination in backtracking searches."
- Mathematicians and computer scientists work on this problem. Though it is admiring, the upper bounds and lower bounds that mathematicians have established are not always applicable to those tackling the problem practically and using computers. [cause, e.g., $\gamma(Q_{n+1}) = \gamma(Q_n) + (0 or 1)$]

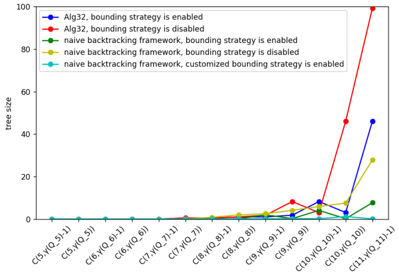
Conclusion and Potential Future Directions

- Despite this, some parts of their proofs or their ideas might be reusable. [req. W.D. Weakley, "Domination in the queen's graph," Graph Theory, Combinatorics, and Algorithms, pp. 1223–1232, 1995.]
- More practical questions might be: max num queens that in an optimal sol can share the same row/ col/ diag? max num queens that can attack each other in an optimal sol?
- Manual proof is possible, but computer-based proof (and potentially more powerful theorems) might also be possible in this particular problem.

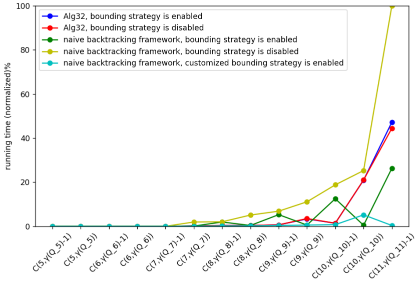
Conclusion and Potential Future Directions (Cont.)



Recall



tree size
Alg32, bounding strategy is enabled
Alg32, bounding strategy is disabled
naive backtracking framework, bounding strategy is enabled
naive backtracking framework, bounding strategy is disabled
naive backtracking framework, customized bounding strategy is enabled
C(5,γ(Q_5)-1)
C(5,γ(Q_5))
C(6,γ(Q_6)-1)
C(6,γ(Q_6))
C(7,γ(Q_7)-1)
C(7,γ(Q_7))
C(8,γ(Q_8)-1)
C(8,γ(Q_8))
C(9,γ(Q_9)-1)
C(9,γ(Q_9))
C(10,γ(Q_10)-1)
C(10,γ(Q_10))
C(11,γ(Q_11)-1)



running time (normalized)%
Alg32, bounding strategy is enabled
Alg32, bounding strategy is disabled
naive backtracking framework, bounding strategy is enabled
naive backtracking framework, bounding strategy is disabled
naive backtracking framework, customized bounding strategy is enabled
C(5,γ(Q_5)-1)
C(5,γ(Q_5))
C(6,γ(Q_6)-1)
C(6,γ(Q_6))
C(7,γ(Q_7)-1)
C(7,γ(Q_7))
C(8,γ(Q_8)-1)
C(8,γ(Q_8))
C(9,γ(Q_9)-1)
C(9,γ(Q_9))
C(10,γ(Q_10)-1)
C(10,γ(Q_10))
C(11,γ(Q_11)-1)

Recall (Cont.)

algo	n	gamma	bounding_strategy_type	tree_size	time_seconds
naive_backtracking_based_on_Bird_thesis	10	4	custom_bounding_strategy	331218	31
naive_backtracking_based_on_Bird_thesis	10	5	custom_bounding_strategy	570407	58
naive_backtracking_based_on_Bird_thesis	11	4	custom_bounding_strategy	150437	14
naive_backtracking_based_on_Bird_thesis	11	5	custom_bounding_strategy	1834480	67
naive_backtracking_based_on_Bird_thesis	12	5	custom_bounding_strategy	15580478	809
naive_backtracking_based_on_Bird_thesis	12	6	custom_bounding_strategy	19706491	971
naive_backtracking_based_on_Bird_thesis	13	6	custom_bounding_strategy	N/A	N/A
Algorithm32	10	4	bounding_strategy_3_1	2719443	22
Algorithm32	10	5	bounding_strategy_3_1	12336135	88
Algorithm32	11	4	bounding_strategy_3_1	4529864	47
Algorithm32	11	5	bounding_strategy_3_1	68919431	289
Algorithm32	12	5	bounding_strategy_3_1	148447295	652
Algorithm32	12	6	bounding_strategy_3_1	298853678	1193
Algorithm32	13	6	bounding_strategy_3_1	N/A	N/A

Recall (Cont.)

algo	n	gamma	bounding_strategy_type	tree_size	time_seconds
Best_of_naive_Algo32	10	4	custom_bounding_strategy	331218	22
Best_of_naive_Algo32	10	5	custom_bounding_strategy	570407	58
Best_of_naive_Algo32	11	4	custom_bounding_strategy	150437	14
Best_of_naive_Algo32	11	5	custom_bounding_strategy	1834480	67
Best_of_naive_Algo32	12	5	custom_bounding_strategy	15580478	652
Best_of_naive_Algo32	12	6	custom_bounding_strategy	19706491	971
Best_of_naive_Algo32	13	6	custom_bounding_strategy	N/A	N/A
SAT	10	4	bounding_strategy_3_1	N/A	4
SAT	10	5	bounding_strategy_3_1	N/A	5
SAT	11	4	bounding_strategy_3_1	N/A	5
SAT	11	5	bounding_strategy_3_1	N/A	15
SAT	12	5	bounding_strategy_3_1	N/A	420
SAT	12	6	bounding_strategy_3_1	N/A	24
SAT	13	6	bounding_strategy	N/A	90 min

Recall (Cont.)

algo	n	gamma	bounding_strategy_type	tree_size	time_seconds
SAT	10	4	bounding_strategy_3_1	N/A	4
SAT	10	5	bounding_strategy_3_1	N/A	5
SAT	11	4	bounding_strategy_3_1	N/A	5
SAT	11	5	bounding_strategy_3_1	N/A	15
SAT	12	5	bounding_strategy_3_1	N/A	420
SAT	12	6	bounding_strategy_3_1	N/A	24
SAT	13	6	bounding_strategy	N/A	90 min
Best_Vals_in_Bird_thesis	10	4	bounding_strategy_3_1	?	<0.014
Best_Vals_in_Bird_thesis	10	5	bounding_strategy_3_1	?	0.014
Best_Vals_in_Bird_thesis	11	4	bounding_strategy_3_1	?	<0.01
Best_Vals_in_Bird_thesis	11	5	bounding_strategy_3_1	?	0.01
Best_Vals_in_Bird_thesis	12	5	bounding_strategy_3_1	?	<0.166
Best_Vals_in_Bird_thesis	12	6	bounding_strategy_3_1	?	0.166
Best_Vals_in_Bird_thesis	13	6	bounding_strategy	?	<3.412

Recall (Cont.)

- Assuming that the theorem.pdf is correct, SAT + theorem, for n=13, and gamma=6 needs about 20 minutes
- But it still struggles with the non-existence cases of $n > 13$

Empty

Next Week

- Degrees, Deadlines, short trip [Q. apply only/others?]
- Might retake, but w/o prep [show pre-loaded link to TOEFL req @ uwindsor link]
- But reading papers and presentations + continuing work on Bird's thesis [Can we have their prog?]
- State-of-the-art SAT solvers did not work better than maplesat
- Linux/C [why postponed applying for cloud] [current skills vs developing ones]
- e.g., Satisfiability: Theory, Practice, and Beyond Boot Camp, SAT-Solving by Armin Biere [figuring out which one best suited here? Backtracking, DPLL, CDCL]

References I

[1] W. H. Bird, "Computational methods for domination problems," Doctoral dissertation, 2017.