New Results on the Queens n2 Graph Coloring Problem

Michel Vasquez

New Results on the Queens_n^2 Graph Coloring Problem

MICHEL VASQUEZ
URC-EMA-CEA, LGI2P Research Center, Parc Scientifique Georges Besse, 30035 Nîmes Cedex 1, France*
email: vasquez@site-eerie.ema.fr

Abstract

For the Queens_n^2 graph coloring problems no chromatic numbers are available for $n > 9$ except where n is not a multiple of 2 or 3. In this paper we propose an exact algorithm that takes advantage of the particular structure of these graphs. The algorithm works on the independent sets of the graph rather than on the vertices to be colored. It combines branch and bound, for independent set assignment, with a clique based filtering procedure. A first experimentation of this approach provided the coloring number values ranging for $n = 10$ to $n = 14$.

Key Words: queens graphs, graph coloring, independent sets, cliques based filtering

1. Introduction

Given a $n \times n$ chess board, a queen graph is a graph with n^2 vertices, each of them corresponding to a square of the board. Two vertices are connected by an edge if the corresponding squares are in the same row, column, or diagonals (*both ascending and descending diagonals*), this corresponds to the rules for moving the queen in a chess game.

The coloring problem for this graph consists in finding the minimum number of colors necessary for placing n^2 queens on the board so that two queens of the same color cannot attack each other. Finding this number (*the chromatic number* χ) is an optimization problem. We may also consider the following decision problem: given a n^2 chess board, is it possible to place n sets (*each of them corresponding to a given color*) of n queens on the board so that there is no clash between two queens in the same set? Gardner (1995) states without proof that this is the case if and only if n is not divisible by 2 or 3. If so, $n = \chi(\text{Queens\_}n^2)$—*noted* χ_n—since the maximum clique number is n (the rows, the columns and the 2 main diagonals constitute the $2n + 2$ maximum cliques of this graph).

Although the graph coloring problem has been the subject of intense research, applications to the Queens-n^2 problem are much scarcer: Mehrotra and Trick (1996) use a column generation approach to the independent set formulation of the graph coloring problem, devising an efficient algorithm tosolve the maximum weighted independent set problem

*Joint "Ecole des Mines d'Alès, Commissariat à l'Energie Atomique" Research Team.

arising in the column generation process, and are able to solve problems up to $n = 9$. Caramia and Dell'Olmo (2001) suggest a sophisticated algorithm based on the iterative coloring extension of a maximum clique; extensive computational results are given, Queens_n^2 problems are solved up to $n = 9$. Heuristic methods canalso be used: Kochenberger et al. (to appear) transform general binary integer problems into unconstrained quadratic binary problems, and solve these problems using the tabu search method. Queens_n^2 problems up to $n = 10$ are tackled, but non-exact methods (see Chiarandini and Stützle, 2002; Hamiez, 2002 for other recent works) fail to prove that $\chi_n = n$ and only give an upper bound for the chromatic number.

2. Our approach

Due to the above observation concerning maximum cliques, the chromatic number of the Queens_n^2 is greater than or equal to n. We consider the question: "*Are n colors enough to color the Queens_n^2?*". Considering this decision problem, and taking into account the specific characteristics of the Queens_n^2 graphs, we shall design a straightforward algorithm.

This algorithm is mainly based on the notion of an independent set (*IS*). An *IS* is a subset of vertices not linked by an edge of the graph: the vertices of an *IS* can all have the same color.

If n colors are enough to color the Queens_n^2 graph, then each coloring with n colors defines n independent sets (*one for each color*). Since no *IS* can contain more than n vertices while n^2 vertices have to be colored, the answer to the above question is positive if and only if there are n disjoint independent sets $\mathcal{I}_1 \ldots \mathcal{I}_n$ with exactly n vertices each.

2.1. *Independent sets*

Hence, the frst step for solving Queens_n^2 consists in enumerating all *IS* with n vertices (in other words in computing the set IS_n of candidate independent sets: the global set of IS). This step is completed by a standard depth-first search that uses the forward checking technique effciently to reduce the search space. Starting from each square of the first row on the chess board, this algorithm erases the column and diagonals under the current row before trying to find a free (*unerased*) square in the next row. A new *IS* is added to IS_n if there is a square remaining in the last row of the chess board.

We thus replace the $vertex \leftarrow \{colors\}$ assigning problem by the $IS_n \leftarrow \{0, 1\}$ assigning problem: the value is 1 if the corresponding *IS* is selected, 0 otherwise. That leads us from a $n^{n \times n}$ search space size to one of size $2^{|ISn|}$. For instance, this preprocessing step applied to Queens_5^2 decreases the number of combinations from 298023223876953000 to 1024 since $|IS_5| = 10$.

Actually, trying to solve the problem by selecting such *IS* avoids manyincorrect color assignments during the coloring process.

2.2. *Branching and backtracking rules*

The choice of these n *IS*, among the $|IS_n|$ candidates, is submitted to the non overlapping constraint: $\forall i \neq j, \mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ (*only one queen by square*). This assignment task is achieved by a branch and bound procedure. Note that assigning one *IS* corresponds to coloring n vertices (*or squares*): the depth of the search tree is less than n.

Every time an *IS* $\mathcal{I}_i$ is selected, propagation on the constraint above is carried out: all $\mathcal{I}_j$ such that $\mathcal{I}_i \cap \mathcal{I}_j \neq \emptyset$ are removed from the global set of *IS:* IS_n.

At every stage in the search we can identify the subsets IS_{ij} of *IS* that cover each free square (i, j) of the chess board. The non colored square which can be covered by the smallest number of remaining *IS* corresponds to the branching node. More precisely, the next branching node is made up of the minimal subset $IS_{\min} = \text{argmin}\{|IS_{ij}|, \textit{(i, j) non colored square}\}$. Exploring this node consists in sequentially selecting each *IS* of $IS_{\min}$. The search tree is not a binary one but a variable breadth one depending on $|IS_{\min}|$.

Eventually, the process backtracks as soon as $IS_{\min} = \emptyset$ occurs.

2.3. *Dynamic filtering based on cliques*

It is not a new idea to reduce the search space while exploring it (see, for instance, the study of Sabin and Freuder (1994) for general CSP framework and Caramia and Dell'Olmo (2002) for a graph coloring application of constraint propagation). The key point of such a technique is to find a condition that eliminates numerous values while needing little computing time to be evaluated.

The principle of our filtering procedure is as follows. After the ith *IS* assignment $n - i$ other *IS* have to be chosen to constitute a solution. If, at this stage of the search, there exists, in the subgraph of non colored squares, a clique of size $n - i$, C_{n-i}, then all the remaining *IS* to be chosen must cover one vertex in this clique. This means that if the condition $\mathcal{I}_j \cap C_{n-i} = \emptyset$ holds for an $\mathcal{I} \in IS_n$, then we can remove this $\mathcal{I}_j$ from the search space under the current node of the tree search. By construction, the $\mathcal{I}_j$ can not produce such a condition with squares belonging to the same row or column (there are no more than n rows and n columns in the chess board, and each $\mathcal{I}_j$ counts n vertices). This is not the case for the diagonals. For example, at the root of the search tree, we can delete the *IS* which do not cover one square of each of the 2 main diagonals. At the next node we can consider these 2 main diagonals plus the 4 with $n - 1$ squares, and so on. We only need to update one counter and implement one test for each diagonal to check this condition.

To summarize, this filtering procedure is based on cliques corresponding to the diagonals of the chess board.

3. Experimentation

In this section we briefly give some details of the largest structures used to implement the algorithm before showing the main results it has produced.

3.1. *Implementation*

The algorithm is written in the *C* programming language.

Independent sets are stored in a global two dimensional array *Big_IS_Array* that contains $|IS_n| \times n$ entries to memorize the n column numbers that define each *IS* in IS_n.

Two other tables of size proportional to $|IS_n|$ are used to keep track of the non overlapping constraint on the remaining *IS*. This constraint is propagated effciently by means of an $n \times n$ array of pointers to tables which contain the addresses in *Big_IS_Array* of the *IS* covering the square in row i and column j. The numbers $|IS_{i,j}|$ of these *IS* are stored in a $n \times n$ array of integers (see figure 1 for an example).

56	36	48	69	63	63	69	48	36	56
36	52	67	51	66	66	51	67	52	36
48	67	66	43	48	48	43	66	67	48
69	51	43	56	53	53	56	43	51	69
63	66	48	53	42	42	53	48	66	63
63	66	48	53	42	42	53	48	66	63
69	51	43	56	53	53	56	43	51	69
48	67	66	43	48	48	43	66	67	48
36	52	67	51	66	66	51	67	52	36
56	36	48	69	63	63	69	48	36	56

Figure 1. Distribution of the *IS* at the root of the search tree for Queens_10^2.

These tables are dynamically updated during the exploration of the search tree. To illustrate this point, we give in figure 2 the new values of IS_{ij} after selecting the first *IS* of $IS_{\min}$ at the first node of the search tree. Following the explanations given in Section 2.2, in this instance, $IS_{\min}$ corresponds to the *IS* that cover the square in the second column of the first row on the chess board ($|IS_{\min}| = 36$).

21	$*$	24	31	31	25	27	21	14	27
13	21	32	$*$	34	31	27	28	21	14
23	33	33	21	19	28	18	$*$	27	19
29	24	$*$	25	18	24	24	21	25	31
30	25	14	29	15	21	29	23	$*$	35
33	25	13	28	22	$*$	15	16	34	35
33	27	19	17	25	17	29	23	31	$*$
$*$	19	39	19	24	20	19	30	32	19
17	28	25	21	33	28	$*$	31	22	16
22	19	22	30	$*$	27	33	28	15	25

Figure 2. $|IS_{i,j}|$ after the first assignment of the search process for Queens_10^2.

Considering Queens_15^2 problem which has 1484400 *IS*, the structure *Big_IS_Array* uses 85 MB of memory, then, for each node of the search tree,we need one array for the pointers to the *IS*, this gives a total amount of $85 \times (1 + 15) = 1360$ MB RAM.

3.2. Results

Computation was carried out on a PENTIUM 4 1.7 GHz CPU with 512 MB RAM.

Up to Queens_11^2 the answer is instantaneous. There is no solution for Queens_10^2. Thus $\mathcal{X}_{10} \geq 11$. Since $\mathcal{X}_{11} = 11$ we deduce that $\mathcal{X}_{10} = 11$ (the first 10 rows and the first 10 columns of the Queens_11^2 solution constitute a 11 colors correct assignment for the 10×10 chess board). Exploring the search tree for Queens_12^2 requires less than 2 hours CPU. The enumeration finds 454 solutions proving that $\mathcal{X}_{12} = 12$. The algorithm hence achieved complete-ness for Queens_10^2 and Queens_12^2.

Unfortunately, instances become quickly intractable when n increases (we limited the CPU time to one week). Finding a solution for Queens_14^2 required some parallelization. The 9990 *IS* of the first $IS_{\min}$ set (which corresponds to the first node of the search tree), are distributed on several CPU. The process starting with the 9987th *IS* provided a solution after 472692 seconds of computing time. This result is weaker than completeness but it is enough to prove that $\mathcal{X}_{14} = 14$.

The figure 3 gives *certificates* for both the Queens_12^2 and Queens_14^2 coloring numbers. Each letter represents a color. We can note symmetries on the two above solutions. The *IS* always appear in pairs and are symmetric (horizontally for $n = 12$ and vertically for $n = 14$). It is also the case for the 454 other solutions of Queens_12^2. Exploiting this remark would divide the tree depth by two. However, even if we could generalize this result, it would not allow us to cope with the Queens_16^2 problem since it counts 4543410 pairs of *IS*: that

A	B	C	D	E	F	G	H	I	J	K	L
L	G	K	I	H	C	J	E	D	B	F	A
E	C	F	L	D	K	B	I	A	G	J	H
H	J	I	B	F	L	A	G	K	D	C	E
K	D	A	G	J	H	E	C	F	L	I	B
C	E	L	F	B	D	I	K	G	A	H	J
J	A	G	K	I	E	H	D	B	F	L	C
I	K	E	C	A	G	F	L	J	H	B	D
D	F	H	J	L	B	K	A	C	E	G	I
G	L	D	E	K	J	C	B	H	I	A	F
F	H	B	A	C	I	D	J	L	K	E	G
B	I	J	H	G	A	L	F	E	C	D	K

A	B	C	D	E	F	G	H	I	J	K	L	M	N
F	J	M	B	N	C	D	L	K	G	I	H	E	A
E	G	L	K	D	B	M	I	H	N	J	A	C	F
J	I	H	M	C	E	F	N	A	D	L	B	K	G
N	D	K	L	G	A	J	C	B	H	E	I	F	M
G	F	N	C	M	I	E	A	J	L	B	K	D	H
I	E	A	H	L	K	N	G	F	M	D	C	B	J
M	H	D	E	F	G	B	K	L	I	A	J	N	C
K	L	B	J	I	M	H	D	C	F	N	G	A	E
B	A	G	F	K	D	C	J	N	E	H	M	L	I
C	M	E	I	J	H	L	B	D	A	F	N	G	K
H	K	F	G	A	N	I	M	E	B	C	D	J	L
L	C	I	N	B	J	A	F	G	K	M	E	H	D
D	N	J	A	H	L	K	E	M	C	G	F	I	B

Figure 3. Certificates for $\mathcal{X}_{12} = 12$ and $\mathcal{X}_{14} = 14$.

Table 1. Filtering efficiency and other global indications.

n	$\|V\|$	$\|E\|$	*pp.t.*	$\|IS_n\|$	$\|IS_n^f\|$	$\|sol.\|$	*cmpl.*	*t.t.*
4	16	76	0	2	0	0	yes	0 sec.
5	25	160	0	10	10	2	yes	0 sec.
6	36	290	0	4	0	0	yes	0 sec.
7	49	476	0	40	32	4	yes	0 sec.
8	64	728	0	92	48	0	yes	0 sec.
9	81	1056	0	352	232	0	yes	0 sec.
10	100	1470	0	724	544	0	yes	0 sec.
11	121	1980	0	2680	1744	8	yes	1 sec.
12	144	2596	0	14200	9440	454	yes	6963 sec.
13	169	3328	2	73712	52008	295	no	168 hours
14	196	4186	15	365596	238088	1	no	168 hours
15	225	5180	73	2279184	1484400	?	no	168 hours

means 277 MB RAM for the *Big_IS_Array* and 8×277 MB for the 8 arrays of pointers to *IS* needed by the search tree.

Finally, the Table 1 summarizes the computational results obtained for instances up to $n = 15$. For each Queens_n^2 instance, we show the number of vertices ($|V|$), the number of edges ($|E|$), the preprocessing—for IS_n—CPU time in seconds ($pp.t.$), the number of *IS* before filtering on n cliques: ($|IS_n|$) and after: ($|IS_n^f|$), the number of solutions found ($|sol.|$), the search completeness state ($cmpl.$) and the total CPU time ($t.t.$).

For $n = 4$ and $n = 6$ no branch and bound is needed because there are not enough *IS* to cover the chess board.

This table answers the question: is $\mathcal{X}_n = n$? for values of n up to 14.

4. Conclusion

We have proposed an exact algorithm for solving the Queens_n^2 coloring problems. This algorithm makes major use of a particular characteristic of these graphs: only independent sets with n vertices are useful to answer the question: "*is Queens_n^2 chromatic number equal to n ?*".

This leads us to a straightforward enumeration approach intended to select n independent sets rather than assign colors to n^2 vertices. The branch and bound procedure is reinforced by an efficient filtering technique based on the cliques belonging to the vertices of the graph which are not colored yet. Thus, at each node of the tree, many independent sets are removed, drastically decreasing the size of the remaining search space.

Experimentation provided 3 new results:

$\mathcal{X}$(Queens_10^2) = 11 , $\mathcal{X}$(Queens_12^2) = 12 and $\mathcal{X}$(Queens_14^2) = 14.

That is the qualitative contribution of this work.

Moreover the results for $n = 12$ and $n = 14$ are the same for Queens_$m \times n$ problems ($m \times n$ *chess boards for which* $m \leq n$) since the maximum clique number is still equal to n when $m < n$.

5. Perspectives

Firstly, the investigation of symmetries noted in Section 3.2 should enable to obtain results for larger instances than Queens_14^2. For this purpose, we suggest to modify significantly the data structures used by our first algorithm. More precisely, the concept aims at changing the balance between speed and memory consumption. We strongly believe that this is a promising way to improve our algorithm.

Secondly, the results obtained on Queens_12^2 and Queens_14^2 constitute an example of some instances that are not too large and have never been solved by any heuristic approach (see for instance Chiarandini and Stützle, 2002; Hamiez, 2002; Kochenberger et al., to appear). Here we have an implement of critical analysis and maybe even a perfecting implement for these inexact resolution methods of dificult problems, that we are going to investigate in a near future.

Acknowledgments

We would like to thank the anonymous referees for the valuable and detailed comments which helped to improve the presentation of this paper.

References

Caramia, M. and P. Dell'Olmo. (2001). "Iterative Coloring Extension of a Maximum Clique." *Naval Research Logistic* 48(6), 518–550.

Caramia, M. and P. Dell'Olmo. (2002). "Constraint Propagation in Graph Coloring." *Journal of Heuristic* 8, 83–107.

Chiarandini, M. and T. Stützle. (2002). "An Application of Iterated Local Searchto Graph Coloring Problem." In Mehrotra, A., Johnson, D.S., and Trick, M. (eds.), *Proceedings of the Computational Symposium on Graph Coloringand its Generalizations*. Ithaca, New York, USA, pp. 112–125.

Gardner, M. (1995). *Further Mathematical Diversions*: *The Paradox of the Unexpected Hanging and Others*. Mathematical Association of America.

Hamiez, J.P. (2002). "Coloration de Graphes et Planification de Rencontressportives: Heuristiques, Algorithmes et Analyses." PhD thesis, Universitéd'Angers.

Kochenberger, G., F. Glover, B. Alidaee, and C. Rego. (to appear). "An Unconstrained Quadratic Binary Programming Approach to the Vertex Coloring Problem." *Annals of Operations Research*.

Mehrotra, A. and M.A. Trick. (1996). A Column Generation Approach for Graph Coloring." *INFORMS Journal of Computing* 8(4), 344–354.

Sabin, D. and E. Freuder. (1994). "Contradicting Conventional Wisdom in Con-Straint Satisfaction." In *ECAI'94*, Amsterdam, pp. 125–129.