DATE'16 UNIVERSITY BOOTH

Putting Queens in Carry Chains, № 27

Thomas B. Preußer
Insititute of Computer Engineering, TU Dresden, Germany
thomas.preusser@tu-dresden.de

***Abstract*—The N-Queens Puzzle is a fascinating combinatorial problem. Up to now, the number of distinct valid placements of N non-attacking queens on a generalized $N \times N$ chessboard cannot be computed by a formula. This is surprising as the set of these placements is equivalent to an interesting subset of the possible permutations of a vector of N distinct elements. Specifically, it is exactly those permutations that alter the relative distance of any two vector elements and, hence, perform an especially thorough shuffle.**

While a backtracking search within the space of all $N!$ possible permuations is very expensive, it also yields a perfect, very scalable computational benchmark for parallel platforms. Only massively parallel distributed computations are capable to to compute solutions counts for $N \geq 24$. This demo presents the open-source infrastructure that is backing a currently ongoing effort to compute the next yet unknown solution count $Q(27)$.

I. OVERVIEW

The N-Queens Puzzle is a 150-year-old mathematical riddle whose comprehensive history is very well described by Bell and Stevens along with interesting generalizations and practical applications [1].

The chess rules allow a queen to move an arbitrary distance in a horizontal, vertical or diagonal direction. This implies that each of the N columns and each of the N rows of a completed solution must contain exactly one of the N placed queens. Any one of the $2N-1$ diagonals in both the upward and downward directions may be occupied by at most one queen. The total number of valid solutions is trivially bounded by $N!$. This is the number of possible placements of N non-attacking rooks, which lack the constraining ability to move along a diagonal.

Mathematically, an N-Queens solution can be viewed as a permutation matrix that contains its one (1) entries exactly at the coordinates of the validly placed queens. The permutation matrices that correspond to an N-Queens solution shuffle an input vector especially well:

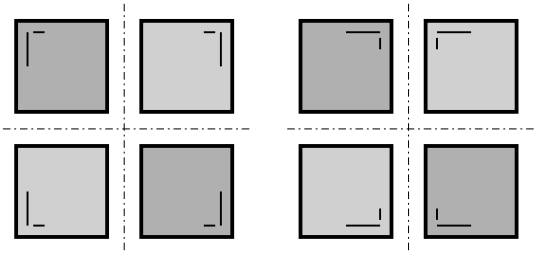
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 1 \\ 4 \\ 2 \end{pmatrix}$$

As no two queens can reside on the same diagonal, the induced permutation does not allow any two elements to maintain their relative postion after the shuffle.

Most interestingly, it is this last constraint that complicates the counting of valid solutions so that the only known approach is the backtracking search. Unfolding the initial levels of the search tree induces a partitioning of the solution space. Its granularity can be easily tuned by the choice of the unfolded depth so that even the huge solution spaces of larger problem sizes can be chopped into *independent* pieces of a manageable size. This way, the N-Queens Puzzle turns into a computational challenge that lends itself as a great show-off case for massively parallel, high-performance computing. It is, indeed, used as such for grid computations [2], Internet computations [3], supercomputers [4], FPGA [5] and for GPU [6] platforms.

II. Q27 PROJECT

The Q27 Project aims at computing the next, yet unknown solution count for the problem size of $N = 27$. The overall problem was chopped into 2 024 110 796 independent subtasks that are processed individually in a massively parallel, distributed computation. These subtasks have been derived exploiting the elementary eight symmetries of a chess board to cut the search space accordingly:



The computation effort is backed by a distributed infrastructure that has been disclosed as open source[1] to encourage others to join into the effort. We hope to attract both those who just want to see the result and those who want to demonstrate the power of their systems. Currently, it is FPGAs tackling one subtask after the other. A GPU port is currently in the process of optimization.

REFERENCES

[1] J. Bell and B. Stevens, "A survey of known results and research areas for n-queens," *Discrete Mathematics*, vol. 309, no. 1, pp. 1 – 31, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0012365X07010394

[2] "nQueens: n=25," INRIA, France, Website: http://www-sop.inria.fr/oasis/ProActive2/apps/nqueens25.html.

[3] "NQueens@Home (beendet)," Rechenkraft.net e.V., Website: http://www.rechenkraft.net/wiki/NQueens@Home_\%28beendet\\%29.

[4] "MC#," Website: http://www.mcsharp.net/.

[5] "Queens@TUD," TU Dresden, Germany, Website: http://queens.inf.tu-dresden.de/.

[6] T. Zhang, W. Shu, and M.-Y. Wu, "Optimization of n-queens solvers on graphics processors," in *Advanced Parallel Processing Technologies*. Springer, 2011, pp. 142–156.

[1] https://github.com/preusser/q27