Constructing Optimal Golomb Rulers in Parallel

Initial Project Abstract

Submitted by

Amrit Goyal - 160905210

Jenit Jain - 160905070

Sahil Garg - 160905058

Semester - 6
Section - CSE D

Department of Computer Science & Engineering

MANIPAL
INSTITUTE OF TECHNOLOGY

Title

Constructing Optimal Golomb Rulers in Parallel

Research Domain

- CSP - Constraint Satisfaction Problems, Algorithms
- Tree search algorithms
- NP-Hard problems
- Backtracking of OGR-n (optimal golomb ruler of n divisions) problem
- Radio astronomy, information theory and error correction and detection

Abstract

A CSP (Constraint satisfaction problem) is defined by a set of variables and a set of constraints. A set of allowed values (the domain) is associated to each variable. Solving a CSP means finding an assignment for each variable that satisfies all the constraints. Finite Domains constraint solving is now a well established technique. Typically, these problems require extensive computation to find a solution.

A Constraint Satisfaction Problem P is given as a tuple P = (X,D,C,R), where :
– X = {X1,...,Xn} is a set of n variables.
– D = {D1,...,Dn} is a set of n domains where each Di is associated with Xi.
– C = {C1, . . . , Cm} is a set of m constraints where each constraint Ci is defined by a set of variables {Xi1,...,Xini } ⊆ X.
– R = {R1, . . . , Rm} is a set of m relations where each relation Ri defines a set of ni-tuples on Di1 × · · · × Dini compatible w.r.t. Ci.

A CSP can be solved using either of the following ways -
- performing constraints propagation before or during search with a number of different filtering techniques
- improving the search by choosing good variable or value ordering heuristics for the next variable to be instantiated and value to be assigned
- improving the search by a more intelligent backtracking, some look-ahead strategies or a combination of them
- performing subproblems decomposition

The golomb rulers problem is one famous example of a CSP. Intuitively, a Golomb Ruler may be thought of as a specific type of ruler: whereas a typical ruler has a mark at every unit measure, a Golomb ruler will only have marks at some subset of these measures. The distance between any two marks on the ruler is different than any other two marks. Any ruler that possesses this trait is called ‘Golomb’.

More formally, we have that a Golomb ruler is a set,

A = {a_1, a_2, ... ,a_N} where a_1<a_2<...<a_N

having the property that the difference,

a_i - a_j (i < j)

is distinct for all pairs (i,j) .

Golomb rulers have a wide variety of other applications, including radio astronomy and information theory. In radio astronomy astronomers use arrays of radio telescopes in a single line to collect data on particular stars or celestial regions. More information may be extracted if there are multiple difference measurements between the telescopes. By placing the telescopes at the marks of a Golomb ruler the number of distance measurements is maximized.

In information theory, Golomb rulers are used for error detection and correction. Golomb rulers of the same length are used to generate self-orthogonal codes, codes that do not share common differences. Such codes allow for easier error checking since redundancies between codes will show up as errors and may then be resent or corrected.

An optimal Golomb ruler is a Golomb ruler of shortest possible length dependant on the number of marks. G(n) denotes this length for a ruler with n marks. In mathematics, a Golomb ruler (named after Solomon Golomb) is a set of integers starting from zero (x1 = 0 < x2 < x3 . . . < xn), selected such that all their cross differences xij : i ≠ j are distinct. A Golomb ruler is said to be Optimal when it has been formed so that xn (the ruler's length) is as small as possible.

In this project, a construction of the Golomb optimal rulers is studied with a tree search approach. Improvements to the basic algorithm are understood and it is parallelised using a shared memory. The application associated to this approach is written in C using the standard CUDA and MPI libraries. The algorithm will use collaborative mechanisms between processors with effective load balancing.

The problem under discussion is a combinatorial optimisation problem, believed (although not yet proven) to be NP-Hard and even the most efficient parallel algorithms (and their implementations) of today, require years of running time to solve instances with n > 24. Besides exposing the already known embarrassingly parallel nature of the OGR-n problem, our parallelization additionally allows for arbitrary selection of the amount of work assigned to each computational node.

Algorithm

This project aims to develop an application using parallel processing APIs - MPI and CUDA, that can generate an optimal golomb ruler for a particular number of divisions in the most optimum and effective manner with the help of load balancing and distributed memory. The tree search based algorithm is used for this.

The construction of optimal Golomb rulers can be modelized as a tree search problem, with the view to minimize the length of the constructed sequence:

• as the length of an optimal n marks ruler does not exceed 1+2+4+···+2n−2 = 2n−1 −1, it is possible to consider that the root value is 0 and that the values of the nodes are between 1 and 2 n−1 − 2 (see figure 2);
• every leaf of the tree symbolizes a sequence which is a solution if the values are ordered and if all the differences are different.

It is now sufficient to have a walk through the search tree, in depth first, keeping the best observed length, to get a simple sequential algorithm finding an optimal Golomb ruler.
This constructive algorithm belongs to the backtrack algorithms class; to be efficient, any recursive tree traversal must be avoided and the use of arrays as elementary data structure is preferred.

The search space can be reduced by applying some simple construction remarks:

- the best possible length best_length is set to initial_limit = 2^n − 1 at the beginning (or it can be fixed at the execution time)
- when placing the k-th mark at position pos(k), there are still r = n − k remaining marks to be placed, which can't have a length remaining_length(r) less than 1+2+···+r = r(r+1)/2 or than G(r + 1): it induces the following new constraint:
    - pos(k) + max(r(r + 1) , G(r + 1)) < best_length
- mirror solutions can be avoided and thus the search tree reduced, by considering only sequences whose last distance is greater than the first one: if the second mark (first except 0) has already been set with the a value, and in order for the last distance b to satisfy b ≥ a + 1, another constraint has to be introduced :
    - pos(k)+max((r − 1)r , G(r))+a+1 < best_length 2
- as the best observed length decreases, more and more inconsistent values can be ignored.

The **algorithm using a single processor** is -
1. Given positive integer n:
2. Calculate L(n), some lower bound for the length of a Golomb ruler with n marks
3. For each integer K ≥ L(n) :
4. Solve problem instance GR-n, K (find if a golomb ruler is possible with length k for n divisions) - using backtracking search
5. If a Golomb ruler has been found : return found ruler, quit search

The algorithm in a parallel processing interface such as CUDA can be described as -
algorithm CUDA-OGR-SEARCH(n, G[])

```
1. Initialize K according to Equation 1.
2. do
        1. Create P pieces
        2. Launch CUDA kernel; each thread runs Algorithm PIECE-GR-n,k on it's piece
        3. if search space of rulers with length K exhausted
        4. Increase K
3. while Golomb ruler of length K not found
4. — Found Golomb ruler is guaranteed to be optimal
5. Output found OGR
```

Where, equation 1 is -
$K = x[n] \geq L(n) = \max(G(n-1)+1,\ n(n-1)/2+1,\ n^2 - 2n\sqrt{n} + \sqrt{n} - 2)$

And **algorithm PIECE-GR-n,K** is -

```
Given positive integers n, K and rulers start[1, . . . , n], end[1, . . . , n]:
1. Let the first mark fixed at distance zero
2. Let the n-th mark fixed at distance K
3. For each possible configuration of marks between the first and the n-th mark, within piece boundaries:
        4.If a Golomb ruler has been formed:
                5. return ruler, quit search
6. return failure
```

References

- http://www.compunity.org/events/pastevents/ewomp2004/jaillet_krajecki_pap_ew04.pdf

Other references include -

- http://artemis.library.tuc.gr/DT2012-0120/DT2012-0120.pdf
- http://www.math.ucla.edu/~radko/circles/lib/data/Handout-1650-1490.pdf
- https://pdfs.semanticscholar.org/12c9/9d852f32292396a6756023f92ffb50fb8ad2.pdf
- http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2003/JustinColannino/
- https://www.researchgate.net/publication/2532838_Genetic_Algorithm_Approach_-To_The_Search_For_Golomb_Rulers