Finding a Parallel Solution for Near Repeat Analysis

KENNETH MCKANDERS (MOREHOUSE COLLEGE)

MENTOR: DR. HAIHANG YOU

Overview

Introduction: Description of a Near Repeat

Research

Procedures

- Program Descriptions
- Computational Analysis

Discussion of Preliminary Results

Future Work

Acknowledgements

Introduction: What is a Near Repeat?

A Near Repeat occurs when two or more elements in a system can be related through a set of rules

Rules can be anything such as a set distance, difference in time, etc.

Right now, we can consider anything to be an element, such as crimes or cells in the body

The overall goal is to relate elements to each other

- Derive a definite path between the elements to figure out a starting point
- Possibly derive elements that can occur after this chain!

Research: Exploring Near Repeat Occurrences

Traditionally, near-repeats are found by comparing each event to every other event in the series.

The complexity for this is $\boldsymbol{O}\left(\frac{\boldsymbol{n(n-1)}}{\boldsymbol{2}}\right)$.

The results of these comparisons are a list of values showing:

- The amount of near-repeats
- The amount of events inside the specified distance, but not the specified time
- The amount of events inside the specified time, but not the specified distance
- The amount of events outside the specified time and distance

Procedure: The Goals

The primary goal of this project is to calculate the Near Repeats in each test system

- This includes finding:
  - Pairwise Relationships
  - N-wise (greater than 2) Relationships

Secondary goals are to measure the advantages of the application of High Performance Computing

- HPC tests will be run on the Kraken system

Procedure: The Programs

Testing was done in two steps

At each step, several test files were evaluated for the presence of near repeats in the system

- Three different distance scales
- Two different time scales

First Step: Serial Code

For the first half of the summer, a serial Near Repeat solution was developed to analyze the necessity of an HPC solution

The development of this solution yielded the following:

- The best way to compare elements is in the **pairwise** manner
  - $f(i_1, i_2, \cdots, i_r) = \begin{cases} r = 2,\ table\ lookup \\ 2 < r \leq n,\ f(i_1, i_2, \cdots, i_{r-1}) \wedge \bigwedge_{j=1}^{r-1} f(i_j, i_r) \end{cases}$
- Storage issues pop up extremely quickly
- Computer slows down with larger datasets
- HPC solution is definitely required!

Second Step: Parallel Code

The serial solution was reworked, with major adjustments

Storage was revamped:

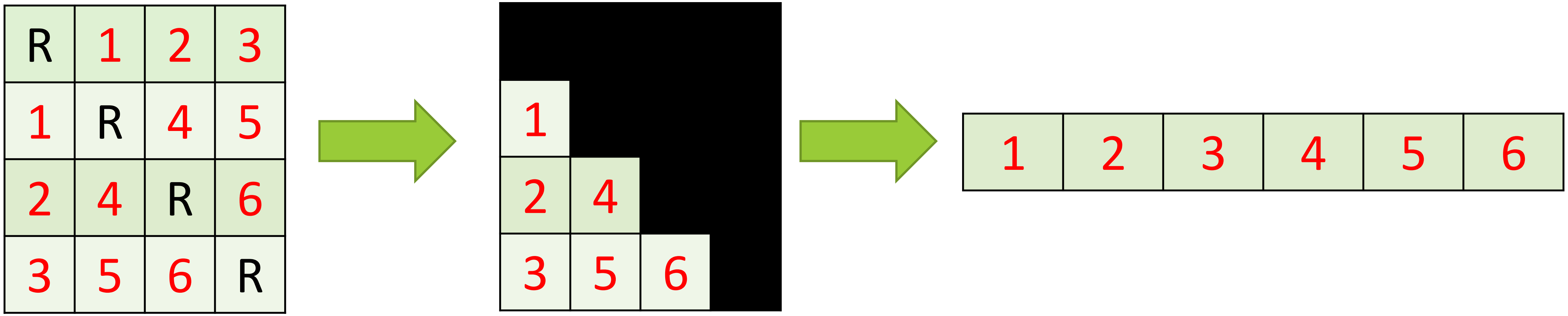
- Traditionally, pairwise comparisons are stored in a 2D matrix ($\boldsymbol{O}(\boldsymbol{n}^2)$)
- Since the matrix is **symmetrical** about the diagonal, and the diagonal contains relations that are each element to themselves, we can eliminate >50% of the matrix from bring stored
- Additionally, the matrix is **sparse**, so we can use **Compressed Row Storage** to eliminate spaces that are not non-zero results (i.e. unrelated elements)

Second Step cont.

This knowledge can be used to optimize the storage space required in the following manner:

1. Eliminate symmetrical spaces
2. Consolidate remaining spaces into a single-dimensional array
3. Compress this array using Compressed Row Storage format

Or, shown visually:



R	1	2	3
1	R	4	5
2	4	R	6
3	5	6	R

1			
2	4		
3	5	6	

1	2	3	4	5	6

Second Step cont.

And finally…

1	3	4	5	6
1	3	1	2	1
0	2	4	5	

Mapping provided by:

$$A_{xy} = \begin{cases} x = y,\ related \\ x \neq y,\ \ Q_a[k]\ \ such\ that\ Q_c[min(x,y)] \leq k < Q_c[min(x,y)+1]\ and \\ Q_b[k] = max(x,y) - min(x,y) \end{cases}$$

Discussion of Current Testing

Small to medium sized files give expected results

Large files throw memory allocation errors, so program needs a little bit of tweaking

HPC solution performs as well for all data sets as the serial code did for smaller datasets

Future Work

Run simulations of larger data sets on Kraken

- This will provide guidelines for the estimation of:
  - Time
  - Cores required

Implement N-wise relation finding in parallel

References

Knox, G. (1964). *The detection of space-time interactions*. Applied Statistics 13:25-29.

Kulldorf, M. and Hjalmars, U. (1999). *The Knox Method and Other Tests for Space-Time Interaction*. Biometrics 55, 544-552.

Acknowledgements

Dr. Haihang You

JICS Department

Oak Ridge National Laboratory

Questions?

Thank you!