Identifying Notable Objects from Spitzer Enhanced Imaging Astronomical Observations

IBhojwani, AlexanderTyan, Sun-Kev, tamos

June 1, 2018

CMSC 12300

The University of Chicago

Goals

- Identify and locate notable objects (outliers) within the sky
- Define an area of interest around that object
- Narrow down search area for manual analysis

Approach

Excess infrared light could mean:

- **Young Stellar Objects**
- Active Galactic Nucleii
- Colliding Galaxies

Hypotheses: Young Stellar Objects

- Notable objects can be identified as extrema in terms of infrared light
- Some notable objects are grouped into interesting structures

Data

- NASA/IPAC Infrared Science Archive:
  - Wide-field Infrared Survey Explorer (WISE)
    - Identifies objects, and readings on the energies they emit
  - 800m objects (records), 815 GB
  - Contains:
    - Location (right ascension, declination)
    - Movement
    - Colour
    - Readings across a number of bands

Unexpected Colour ➡ Interesting Object

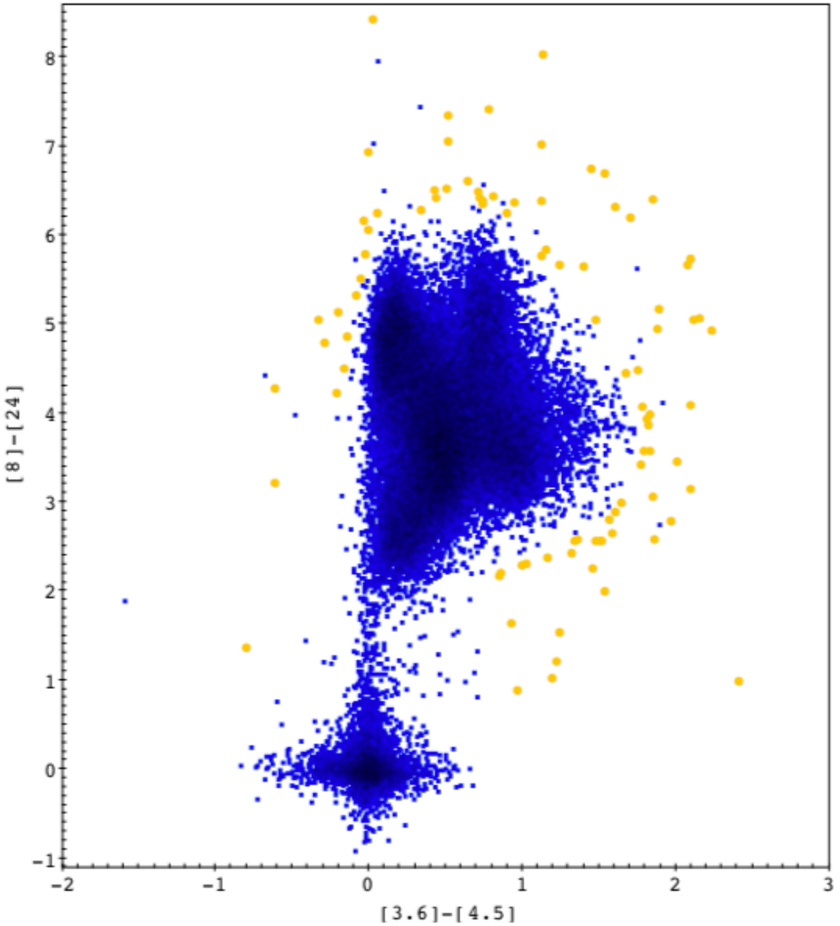


FIGURE 4: Color-color plot of SEIP sources, SNR≥10, not in the galactic plane, and not in major survey areas (blue), vetted color outliers (yellow).

Source: Gorjian et al.

Algorithm I

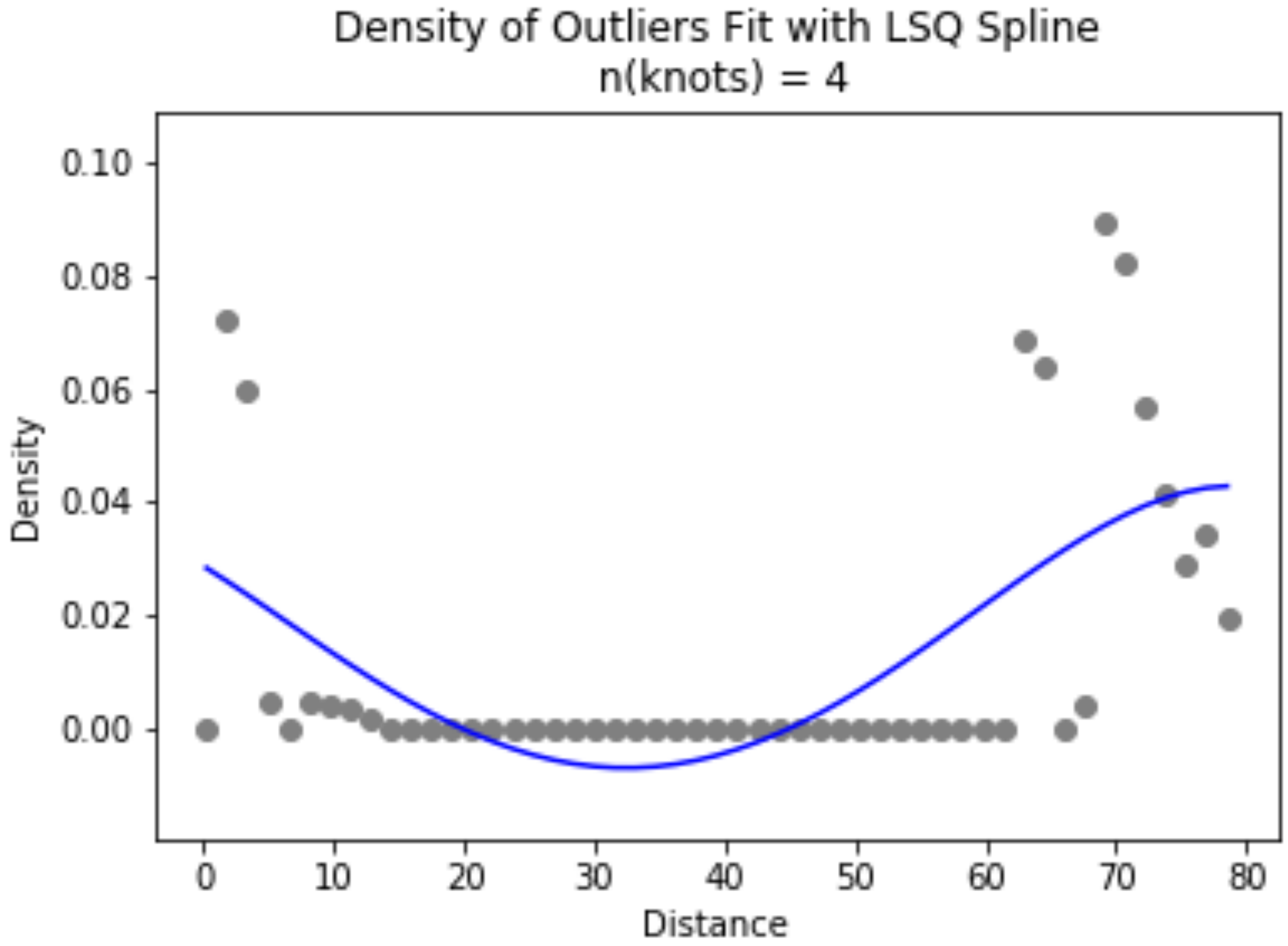
Algorithm I, Step 1: Preprocessing with K-Means Classification

- Canned K-Means in Spark (Dataproc Pending)
- Split points into two groups, ~ N within each

Algorithm I, Step 2: Outliers as Nodes, Distances as Edges

- MRJob
- Within each group, take outliers: x_i: $x_i \geq \mu \pm 2\sigma$
- Developed algorithm with parameters N, P, K.
- Complexity:
  - If number of cases < N(P): Does not run
  - Otherwise, approximately:
    - $(\sum_{i=1}^{(N-N*P)}(N*P+i))*\frac{size-(N*P)}{(N*P)}$
  - Compare to:
    - $size^2$

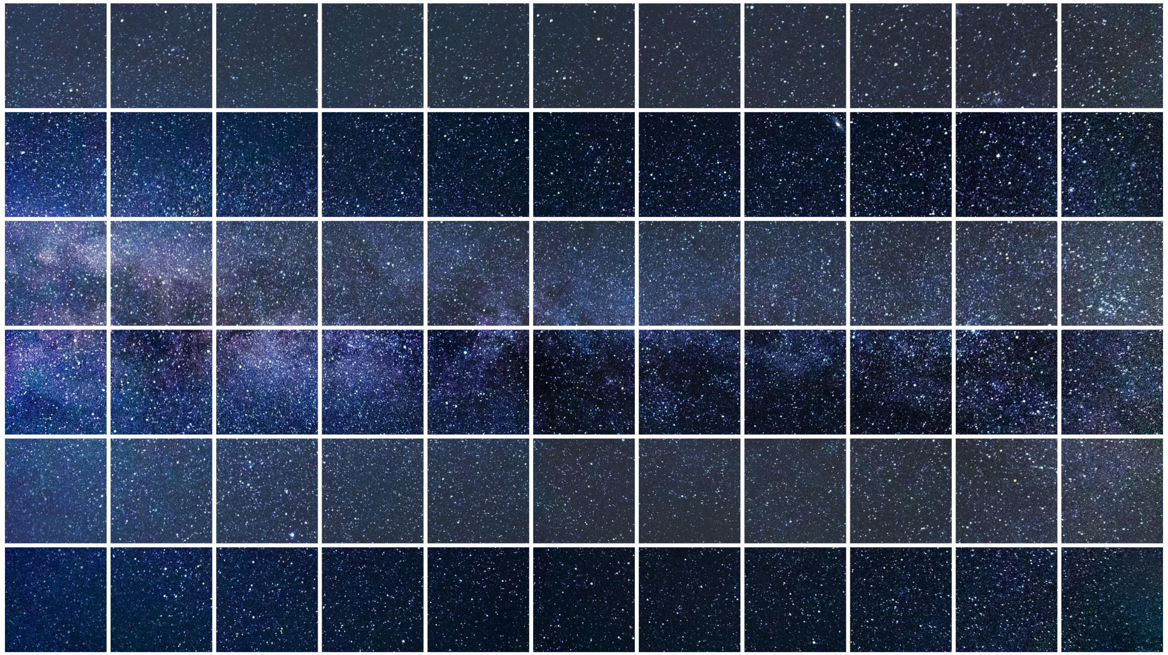
Algorithm I: Distance Density with Fitted Spline



Algorithm II

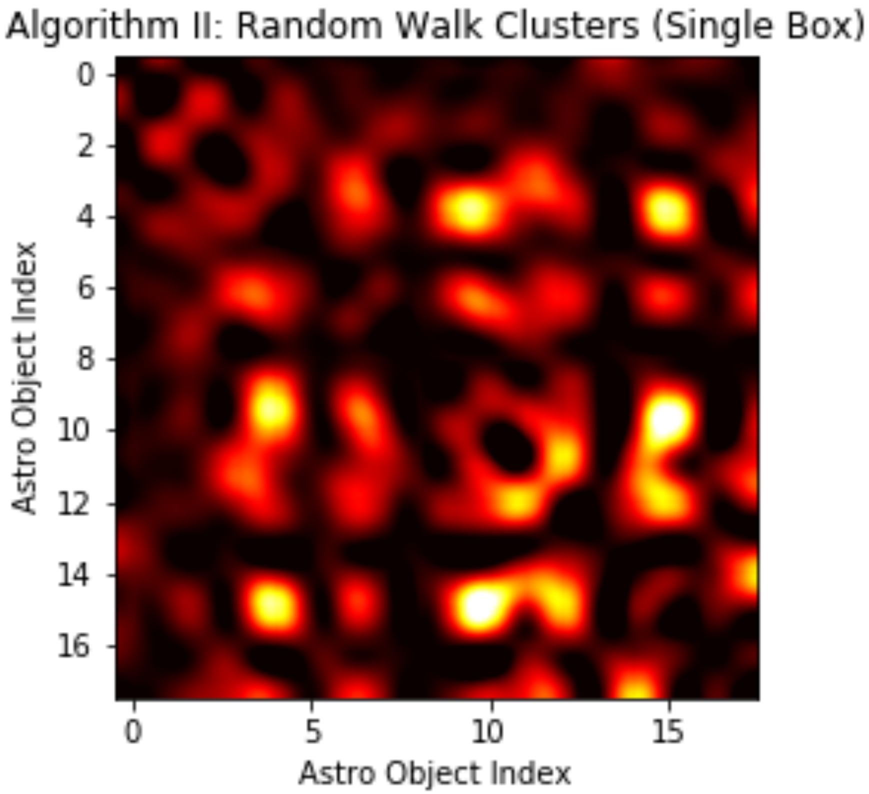
Algorithm II (MapReduce)

- Step 1: Split Sky (Dataproc Implementation in Progress)
  - Create grids of the entire sky
  - Create graphs of each grid



Algorithm II (MapReduce)

- Step 1: Split Sky (Dataproc Implementation in Progress)
  - Create grids of the entire sky
  - Create graphs of each grid
- Step 2: Random Walk
- (Dataproc in Progress)
- Step 3: Find Probabilistic Clusters (Coding In Progress)



Runtime Comparison

Algorithm I

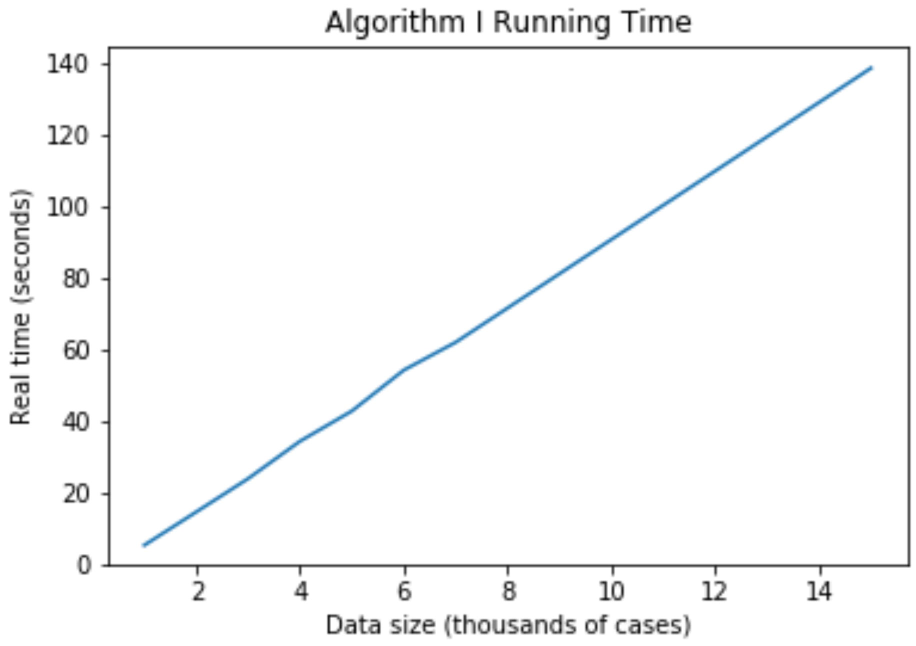
- Local:
- Dataproc:
  - 8 min. 44 sec.
    - 25 workers with n1-standard-4 specs
    - 358,169 rows

Algorithm II (RandomWalk)

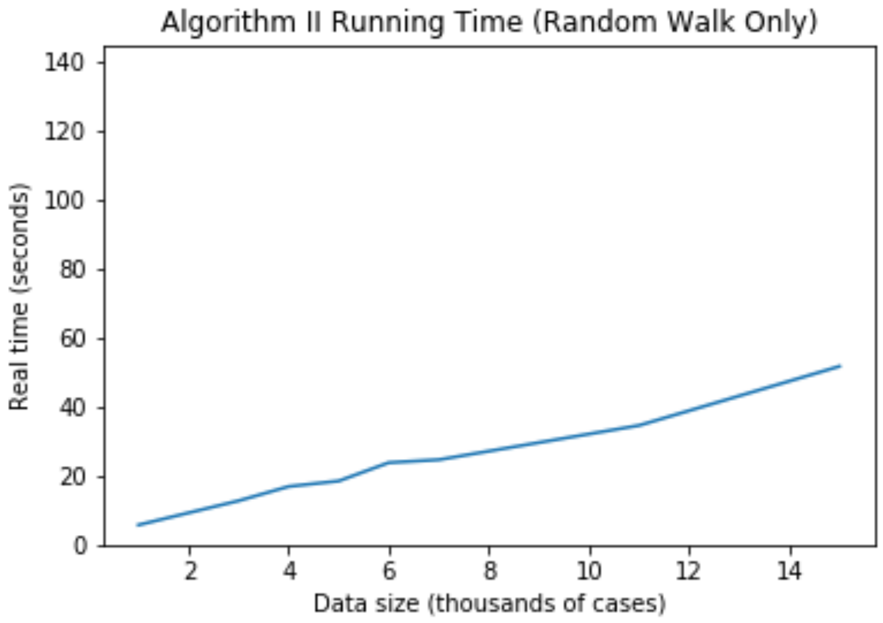
- Local:
  - 16 min. 9 sec.
  - 365,601 rows
- Dataproc:
  - In Progress

Runtime Comparison

Algorithm I



Algorithm II (RandomWalk)



Challenges

- **Problems:**
  - Graph algorithms without all data in memory (sky is too large)
  - Complexity too great for a fully connected graph
- **Solutions**:
  - Running random sample (Algorithm I)
  - Grids and streaming processing (Algorithm II)

Algorithm 1 Implementation

1. **Initialize a node list (Map 1)**
2. **for each object: (Map 1)**
   1. add the object to our node list
   2. if the node list is too large: remove nodes at random until the list is size N(P):
3. **If the node list is sufficently full (size N(P)):**
   1. yield the object's id and distance to each node
4. **For each object (Reduce 1)**
   1. take K edges (the closest K)

Algorithm 1 Implementation (Cont'd)

5. **For each object (Map 2)**
    1. Create a density function of distances from the object
    2. Fit a spline function along this density curve
    3. Find the first saddle point in this spline, usually a local minimum
    4. If the first saddle point is less than 1 return the object and its saddle point

Algorithm 2 Implementation

1. **Grids:**
    1. r x c grids -> bite-sized chunks
2. **Output key: grid id**

Algorithm 2 Implementation (Cont'd)

3. **Random Walk**
    1. Fully connected graphs w/ distance^{-1} as edge weight
    2. Random walks on graph -> gradient of re-visit frequency

Algorithm 2 Implementation (Cont'd)

4. **Watershed Algorithm:**
    4. Segmentation of the "image" - i.e, gradient

Code: https://github.com/ibhojwani/seip_big_data