Linking As-Planned and As-Executed Data in Near-Real Time

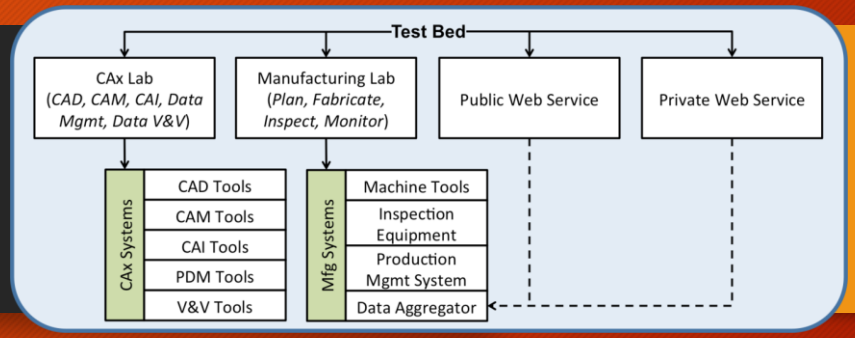
Michael Roa

Moneer Helu

University of Maryland
NIST/Engineering Laboratory/Systems Integration Division

Background

Application to the SMS Test Bed



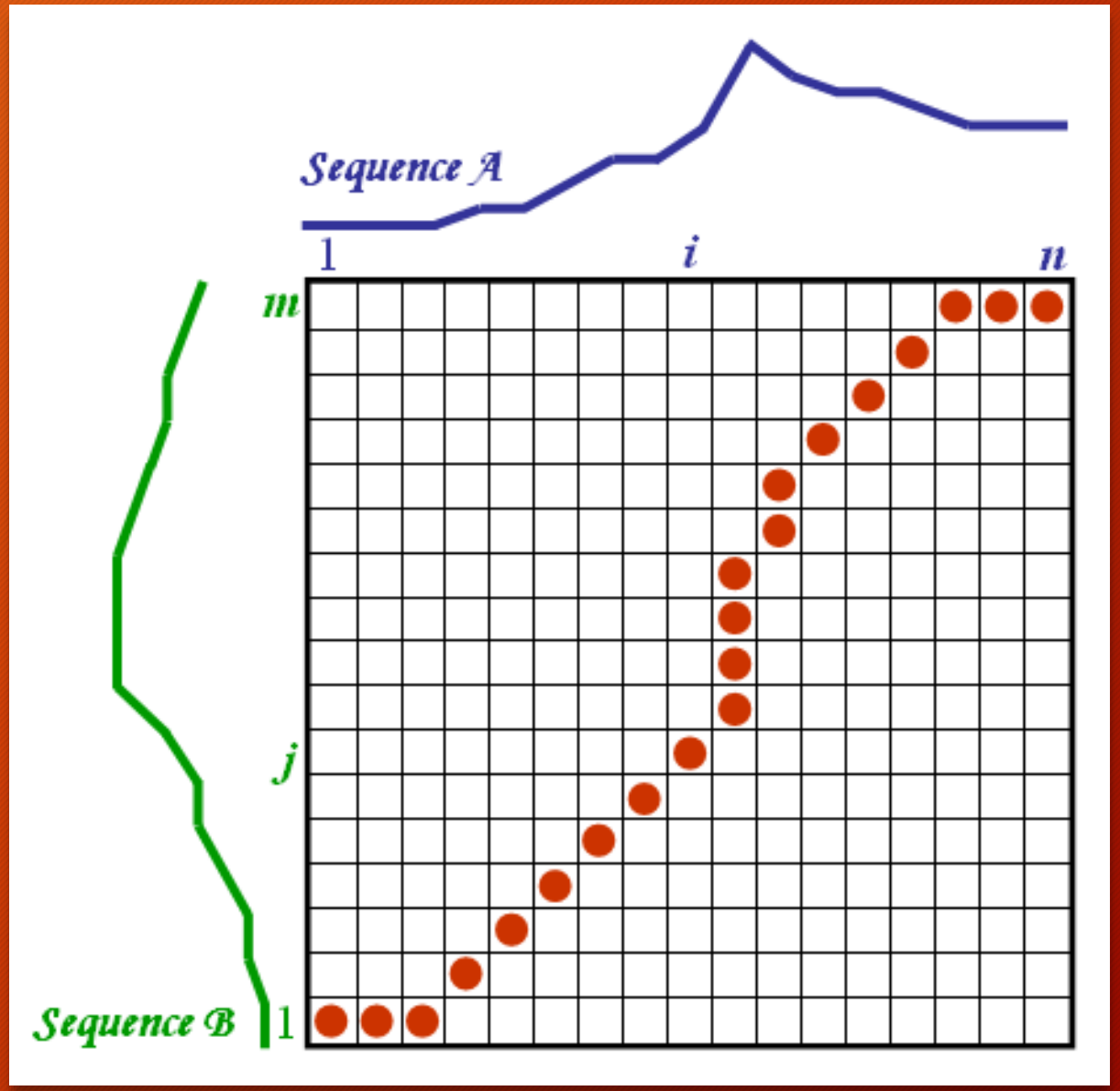
- DTW is used to analyze data from Computer Numerical Controlled(CNC) machines
- This offers valuable insight to machine performance and behavior.
- The problem is current methods offer **no real-time solution.**
- A real-time solution will offer opportunity for immediate feedback to the concurrent processes

Introducing Real-Time DTW

- Introducing a real-time DTW solution can better asses machine behavior
- Real-time solutions will allow decision making closer to the time of execution allows for effective changes early in the execution process
- Current Implementation in R cannot accomplish real-time DTW
  - Code execution is slow, and unwieldly for real-time
  - The mtconnectR package was not designed with real-time analysis in mind
  - R cannot be deployed as a standalone application nor rapidly
- A combination of Java and Windowed Time Warping(WTW) is identified as the most promising environment and method for the project goals

What is Dynamic Time Warping(DTW)?

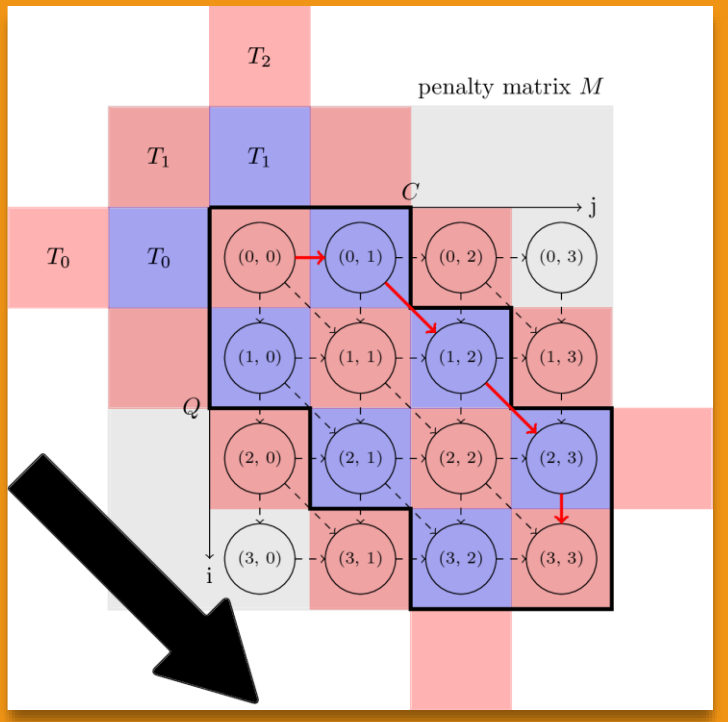
- A technique to identify similarities between two time series data sets
- Commonly used in voice analyzation software, power analysis, and gesture recognition
- Multiple variations of DTW methods; Fast DTW, Windowed DTW, k-nearest-neighbor DTW
- Can help identify discrepancies between a planned and executed process



Research

Investigating Dynamic Time Warping Methods

- DTW has multiple methods and techniques.
- Identifying which of these is most feasible was the first step
  - Traditional DTW is slow and memory hungry; $O(n^2)$
  - FastDTW memory efficient but suited for long-term application; O(n)
  - Windowed Time Warping is short-term, fast, and memory efficient; O(n)
- Windowed Time Warping and FastDTW offered the best advantages for the goals of project



Effectiveness of Fast Dynamic Time Warping (FastDTW)

- Calculates the path at a lower resolution, then uses this path to estimate the higher resolution path
- Looks at small frames of the data, accurately and quickly.
- Uses a Sakoe-Chiba band to cull un-necessary calculations.
- However, with all its advantages FastDTW does not fit the project goals
- FastDTW still requires a full dataset to calculate the path. This would not be real-time.

Effectiveness of Windowed Time Warping (WTW)

- WTW offers a plethora of advantages to common DTW methods
  - Faster than other implementations
  - More accurate than other implementations
  - Does not require a full data set to necessarily work effectively
- Overall WTW offers the most promising results of the known DTW methods for real-time analysis
- This is accomplished by combining the advantages of FastDTW,, and an A-Star pathing implementation

Algorithm

DTW Common Algorithm

Typical DTW implementations use this algorithm

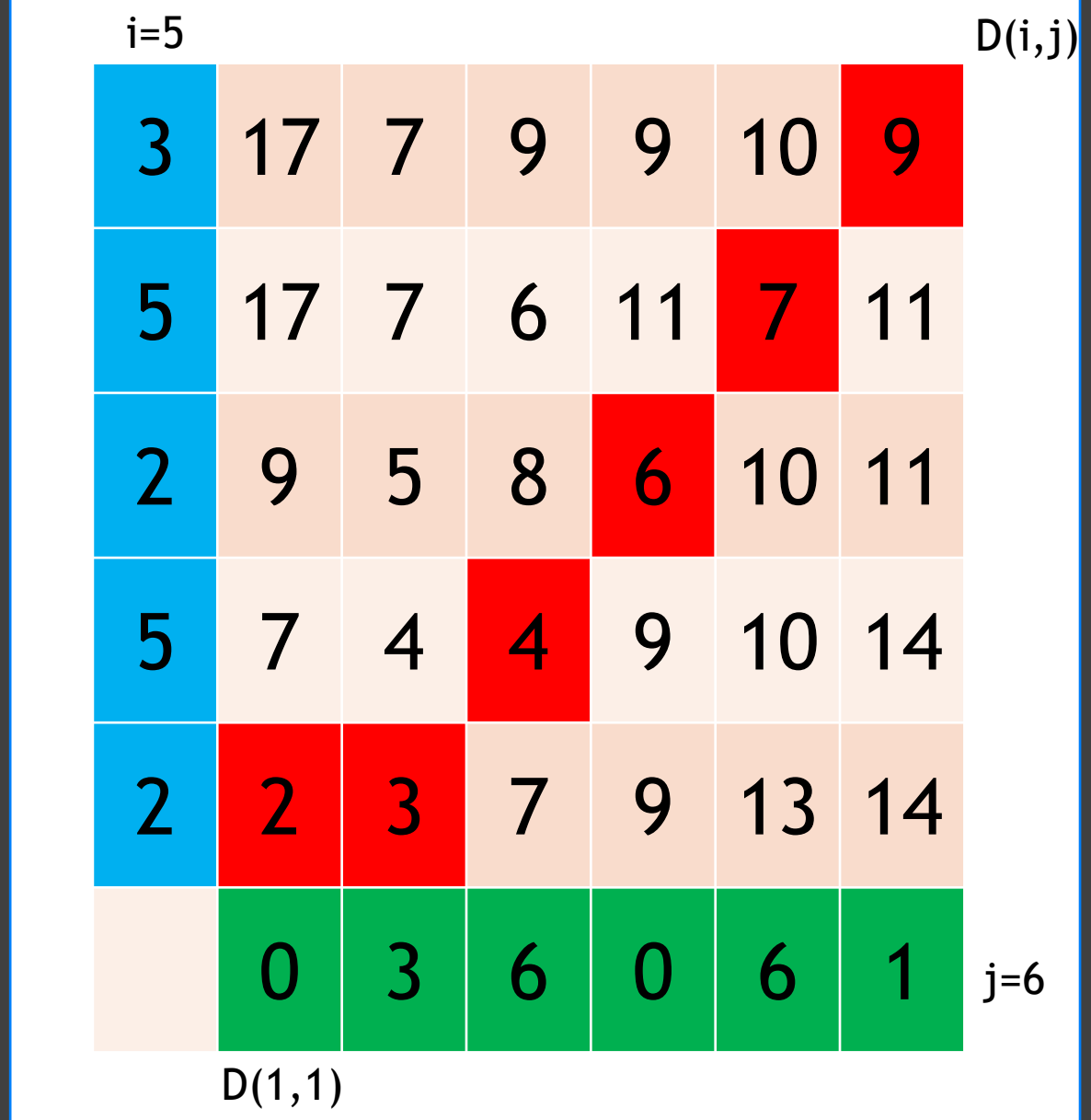
Computes the distance cost of every space in each array

Lowest distance is then used to fill the return array

Complexity is determined by the size of input arrays

$$D(i,j) = |t(i) - r(j)| + min \left\{ \begin{matrix} D(i-1,j) \\ D(i-1,j-1) \\ D(i,j-1) \end{matrix} \right\}$$

1. $1 \leq m_k \leq M$ and $1 \leq n_k \leq N$ for all k.
2. The sequences have endpoints $(m_1, n_1) = (1, 1), (m_L, n_L) = (M, N)$
3. The seqences m_k, n_k are monotonically increasing.
4. $(m_k - m_{k-1}, n_k - n_{k-1})$ must be one of $(1, 0), (0, 1), (1, 1)$.




```
int DTWDistance(s: array [1..n], t: array [1..m]) {
    DTW := array [0..n, 0..m]

    for i := 1 to n
        DTW[i, 0] := infinity
    for i := 1 to m
        DTW[0, i] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],
                                        DTW[i  , j-1],
                                        DTW[i-1, j-1])

    return DTW[n, m]
}
```

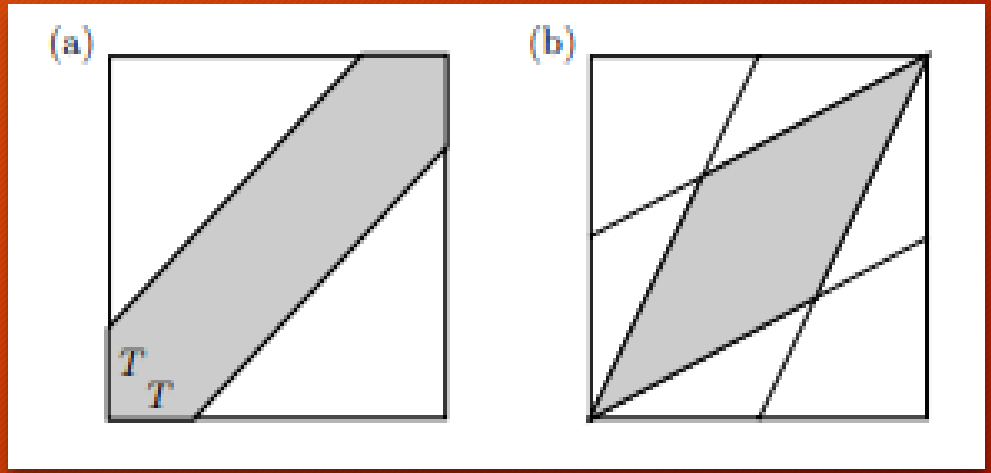
DTW Constraint Algorithm

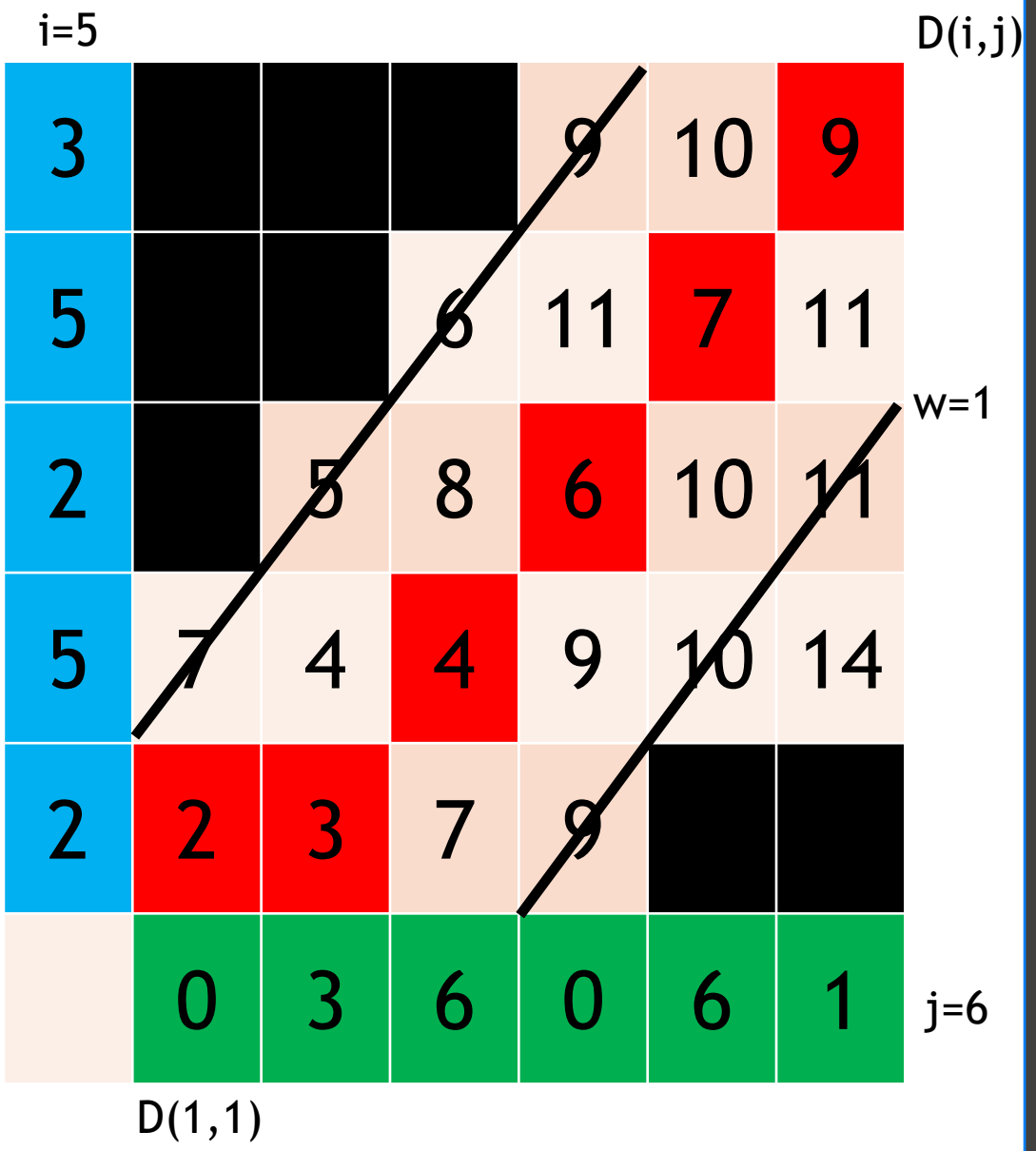
Follows the same general idea as the common algorithm

Constraints are used to restrict computed vectors of the matrix

Improves over common algorithm, but still requires a fairly large set of data

Does not improve complexity






```
int DTWDistance(s: array [1..n], t: array [1..m], w: int) {
    DTW := array [0..n, 0..m]

    w := max(w, abs(n-m)) // adapt window size (*)

    for i := 0 to n
        for j:= 0 to m
            DTW[i, j] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := max(1, i-w) to min(m, i+w)
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                        DTW[i  , j-1],    // deletion
                                        DTW[i-1, j-1])    // match

    return DTW[n, m]
}
```

Windowed Time Warping Algorithm

Significant differences from the common DTW approaches.

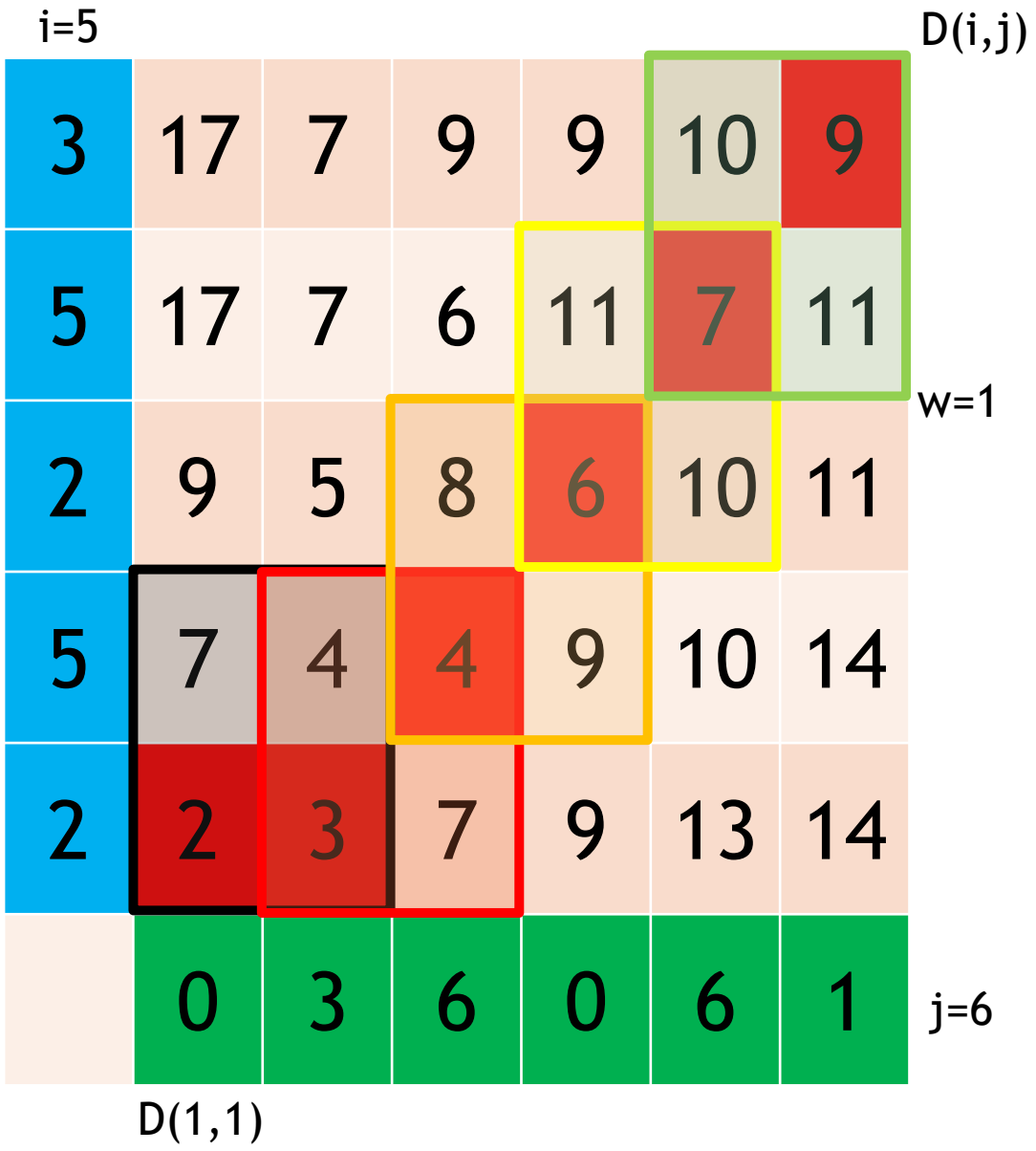
Complexity is O(n)

Accuracy can rival matches made through DTW approaches

Extremely fast compared to DTW

Can utilize smaller frames of data, yet still achieve good accuracy

```
Input: Feature Sequence A and Feature Sequence B
Output: Alignment Path
Path = new Path.starting(1,1);
while Path.length < min (A.length,B.length) do
    Start = Path.end;
    End = Start;
    while (End - Start).length < Window_Size do
        End =
        argmin(Inner_Product(End.next_points));
    end
    Cost_Estimate = End.cost;
    A-Star_Matrix =
    A_Star_Fill_Rect(Start,End,Cost_Estimate);
    Path.add(A_Star_Matrix.getPath(1,Hop_Size));
end
return Path;
```



Input: Feature Sequence A and Feature Sequence B
Output: Alignment Path

```
Path = new Path.starting(1,1);
while Path.length < min (A.length,B.length) do
    Start = Path.end;
    End = Start;
    while (End - Start).length < Window_Size do
        End =
        argmin(Inner_Product(End.next_points));
    end
    Cost_Estimate = End.cost;
    A-Star_Matrix =
    A_Star_Fill_Rect(Start,End,Cost_Estimate);
    Path.add(A_Star_Matrix.getPath(1,Hop_Size));
end
return Path;
```

Conclusion

- From this project I learned:
  - R syntax and operation
  - MTConnect Agent protocol
  - NIST G-Code standard
  - Dynamic Time Warping algorithms
  - Efficient data structure creation
  - Software design skills
- Over the summer I hope to further the develop the program and add a UI, web publishing capabilities, and real-time visualizations

Thank you!

Michael Roa

Moneer Helu & Thomas Hedberg

Montgomery College & NIST, Systems Integration Division