Introduction to Data Association

CSE598C Fall 2012

Bob Collins, CSE, PSU

Outline

TODAY

- Data Association Scenarios
- Track Filtering and Gating
- Global Nearest Neighbor (GNN)
- Review: Linear Assignment Problem
- Murthy's k-best Assignments Algorithm
- Probabilistic Data Association (PDAF)
- Joint Probabilistic Data Assoc (JPDAF)
- Multi-Hypothesis Tracking (MHT)
- Markov Chain Monte Carlo DA (MCMCDA)

Intro to Data Association

Let's consider a different tracking "paradigm".

- Detect objects in each frame.
- Figure out interframe correspondences between them

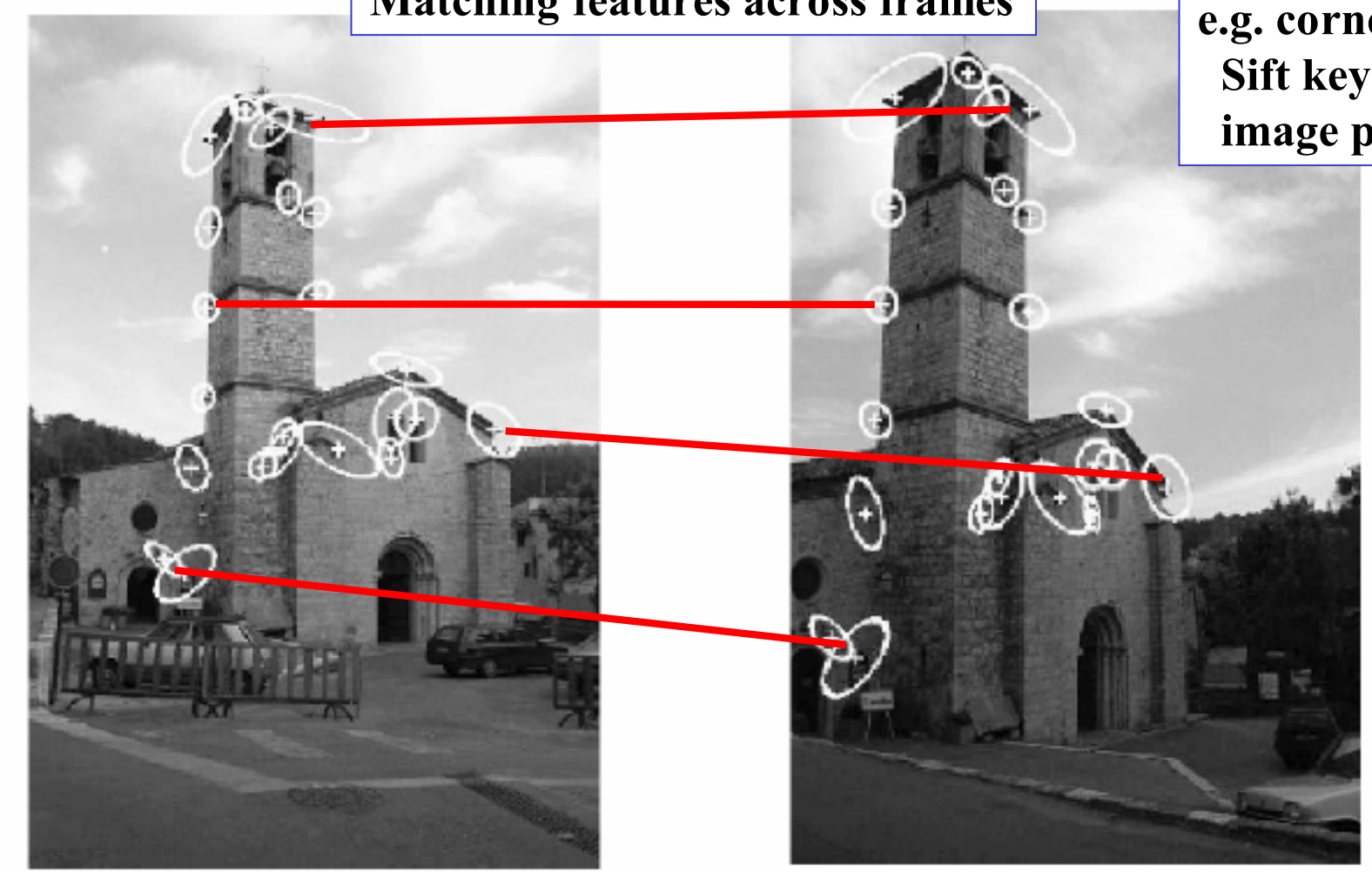
This is a rather natural way of looking at tracking if you are tracking blips on a radar screen. However, this approach also prevails in the domain of automated surveillance systems.

Data Association Scenarios

Two-frame Matching (Correspondence Problem)

Matching features across frames

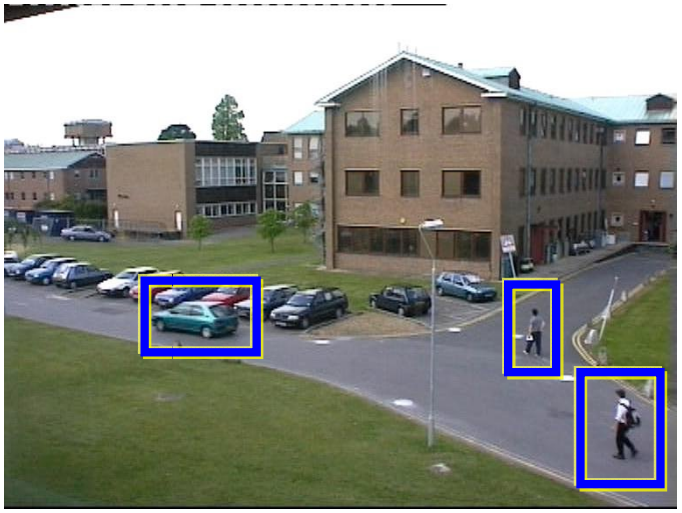
e.g. corners, Sift keys, image patches

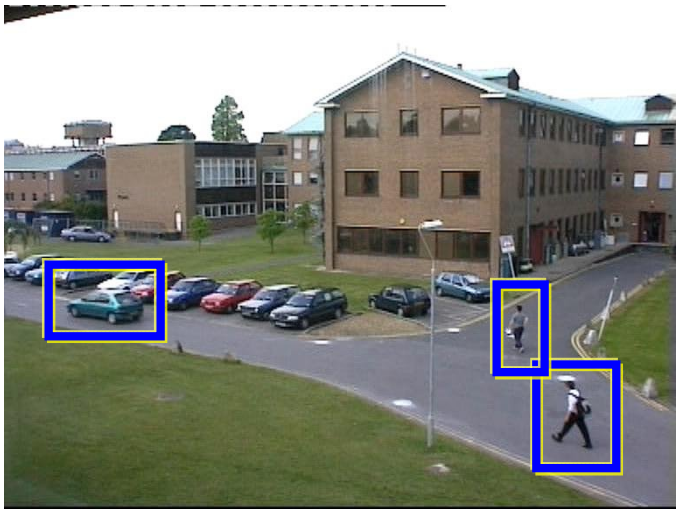


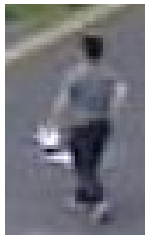
Data Association Scenarios

Two-frame Matching (Correspondence Problem)

Match up detected blobs across frames

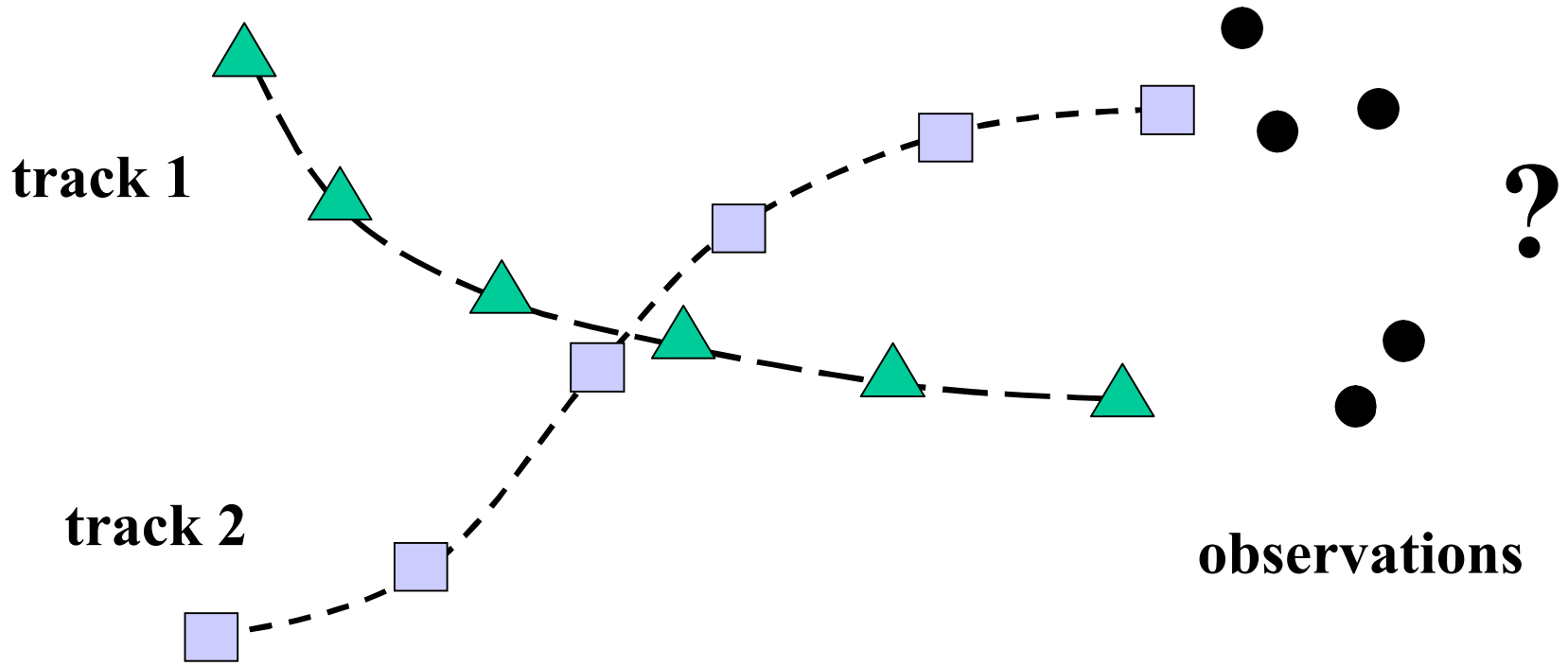






Data Association Scenarios

Multi-frame Matching (matching observations in a new frame to a set of tracked trajectories)



How to determine which observations to add to which track?

Filtering Framework

Recall our earlier discussion of state space filtering

We want to recursively estimate the current state at every time that a measurement is received.

Two step approach:

1) prediction: propagate state pdf forward in time, taking process noise into account (translate, deform, and spread the pdf)

2) update: use Bayes theorem to modify prediction pdf based on current measurement

But which observation should we update with?

Filtering, Gating, Association

Add Gating and Data Association

1) prediction: propagate state pdf forward in time, taking process noise into account (translate, deform, and spread the pdf)

2) Gating to determine possible matching observations

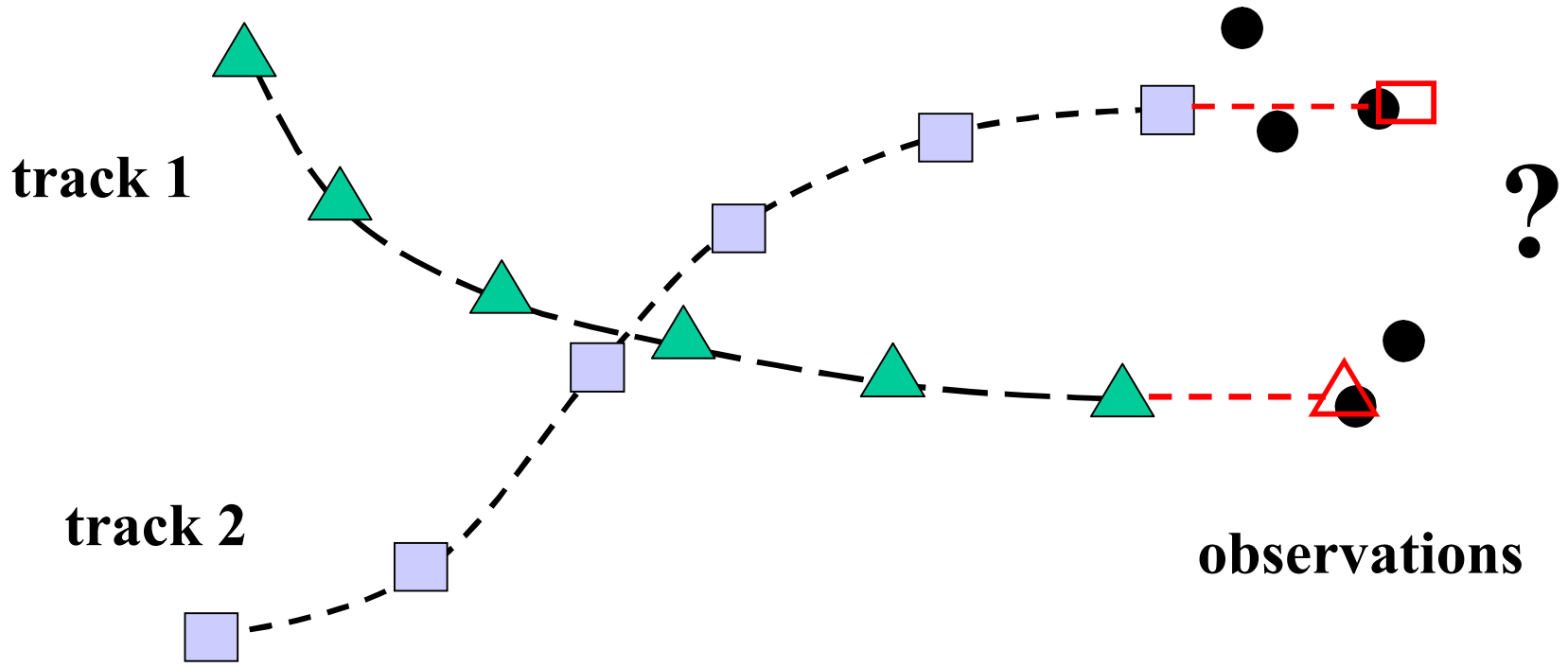
3) Data association to determine best match

4) update: use Bayes theorem to modify prediction pdf based on current measurement

Robert Collins
CSE598C

Tracking Matching

Intuition: predict next position along each track.

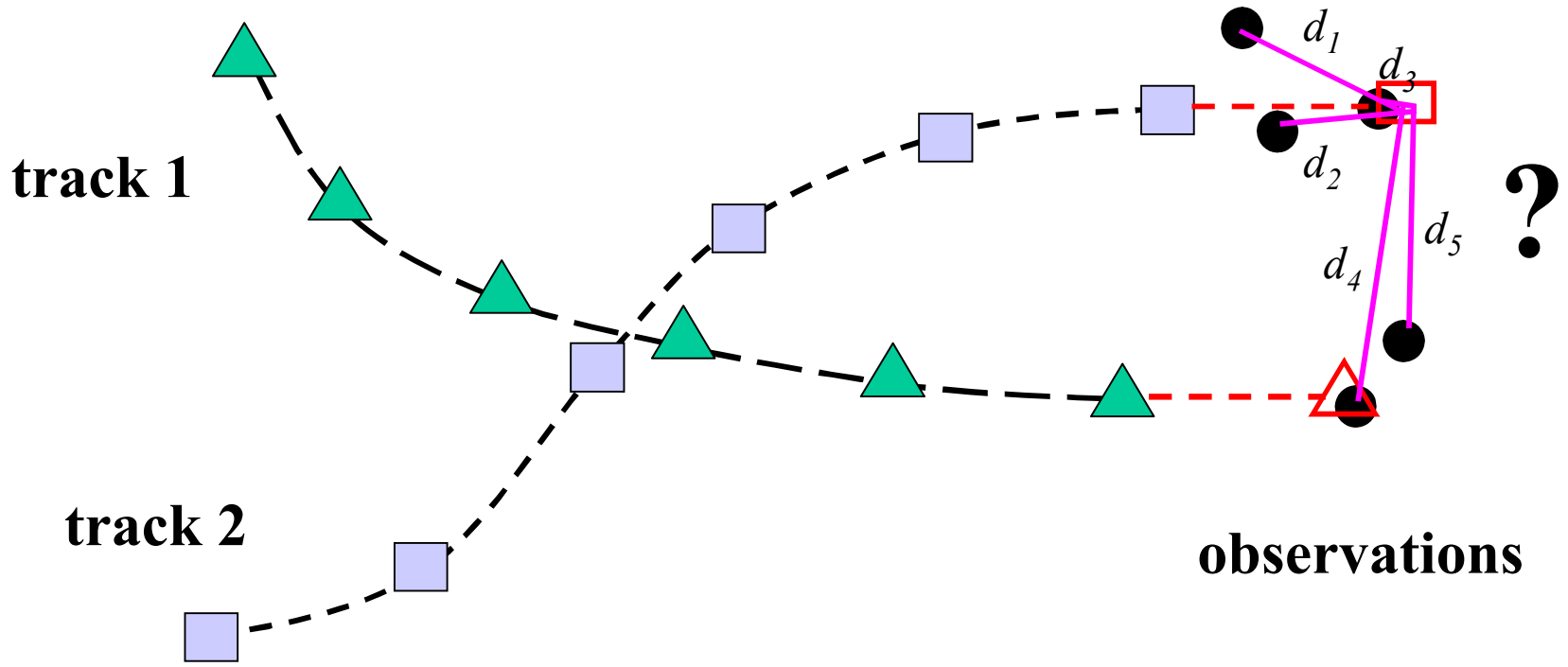


How to determine which observations to add to which track?

Robert Collins
CSE598C

Tracking Matching

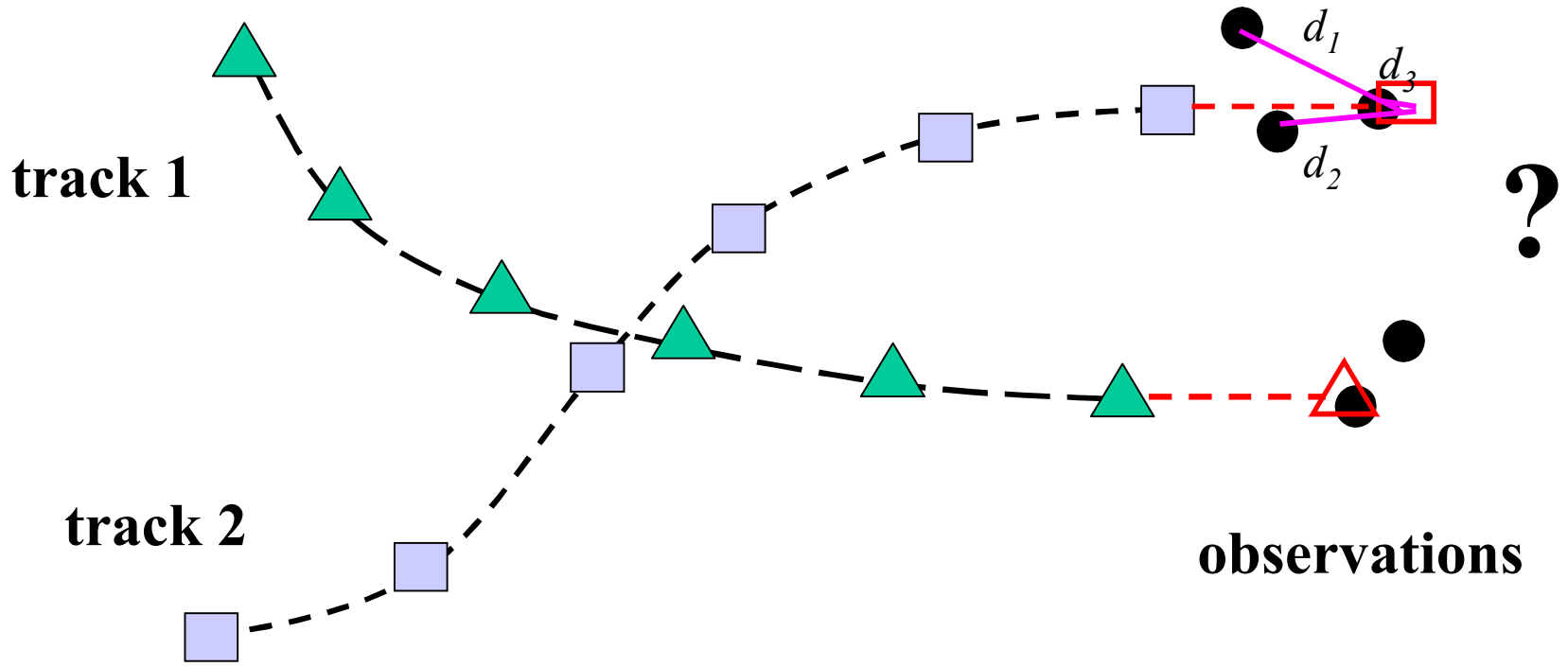
Intuition: predict next position along each track.
Intuition: match should be close to predicted position.



How to determine which observations to add to which track?

Tracking Matching

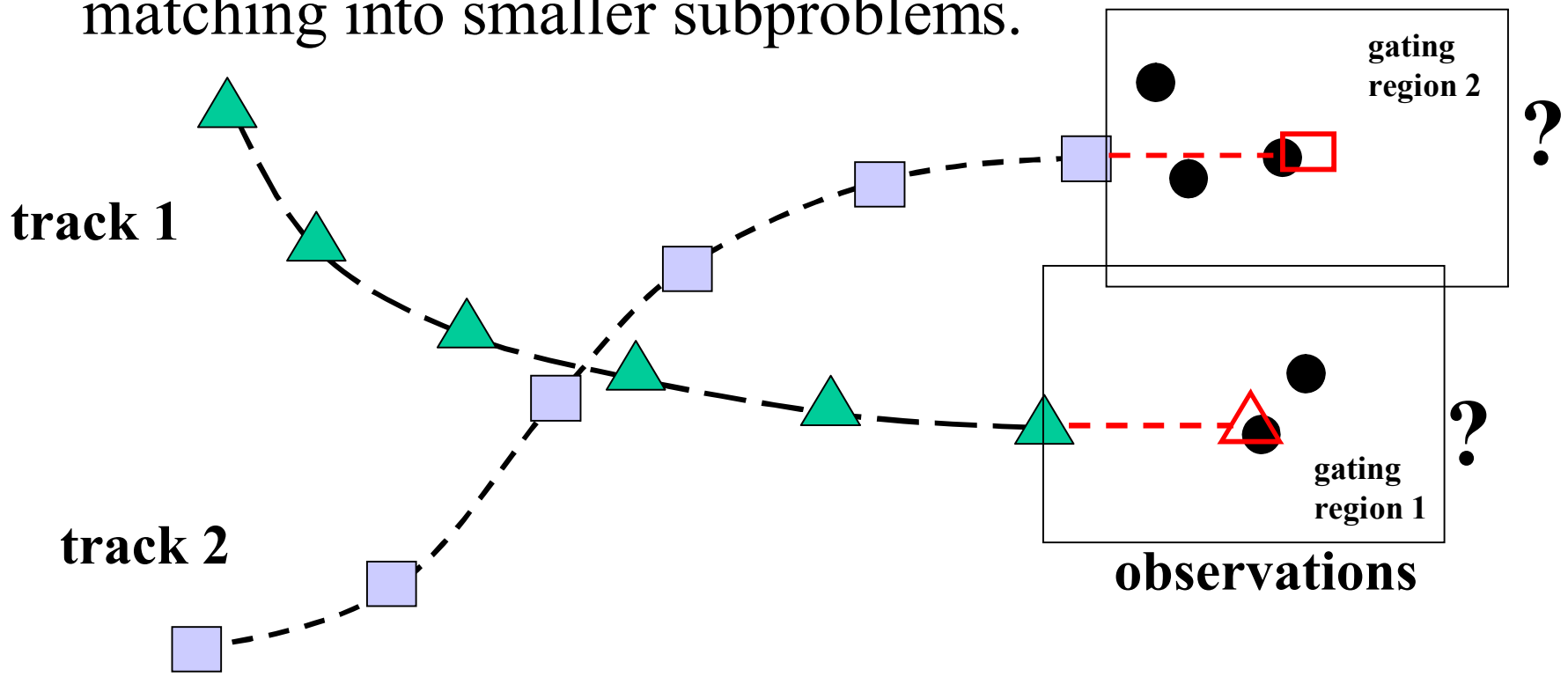
Intuition: predict next position along each track.
Intuition: match should be close to predicted position.
Intuition: some matches are highly unlikely.



How to determine which observations to add to which track?

Gating

A method for pruning matches that are geometrically unlikely from the start. Allows us to decompose matching into smaller subproblems.



How to determine which observations to add to which track?

Recall: Kalman Filter

Predict

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad \text{(predicted state)}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad \text{(predicted estimate covariance)}$$

Update

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad \text{(innovation or measurement residual)}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \quad \text{(innovation (or residual) covariance)}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \quad \text{(Kalman gain)}$$

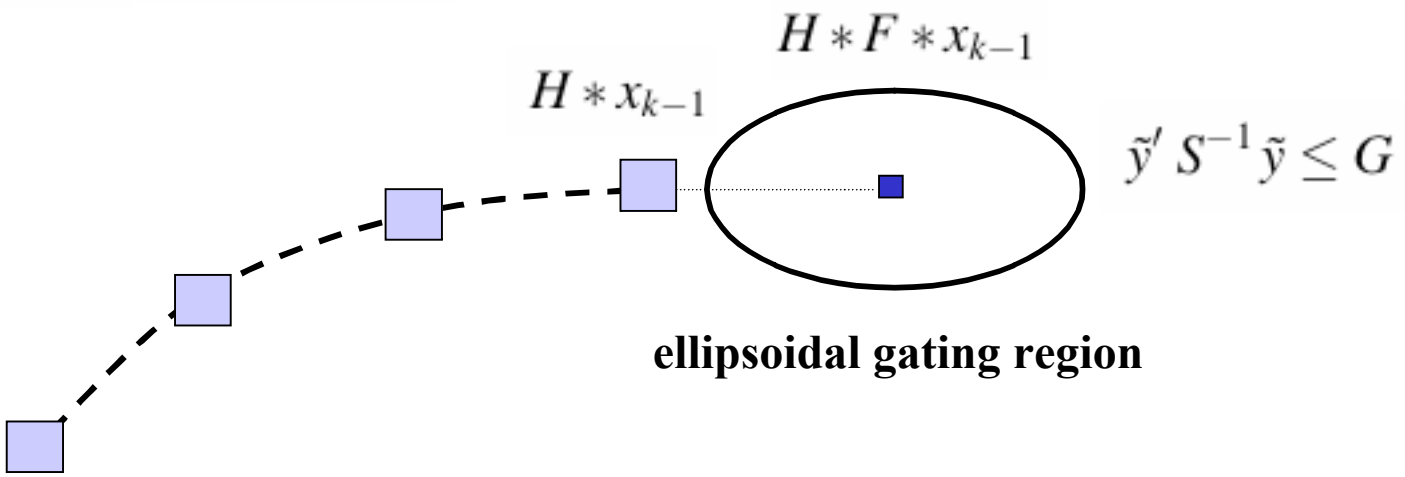
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \quad \text{(updated state estimate)}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \quad \text{(updated estimate covariance)}$$

Robert Collins
CSE598C

Kalman Filter Predict/Gating

$$x = \begin{bmatrix} x \\ y \\ u \\ v \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



ellipsoidal gating region

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad \text{(predicted state)}$$

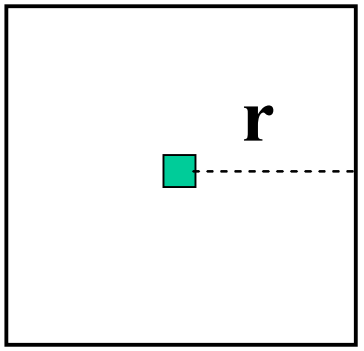
$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad \text{(predicted estimate covariance)}$$

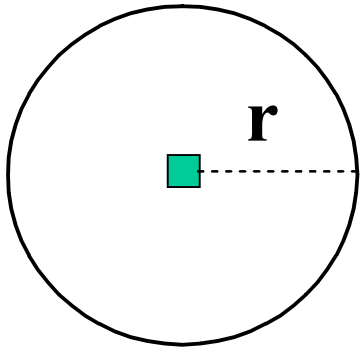
$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad \text{(innovation or measurement residual)}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \quad \text{(innovation (or residual) covariance)}$$

Simpler Prediction/Gating

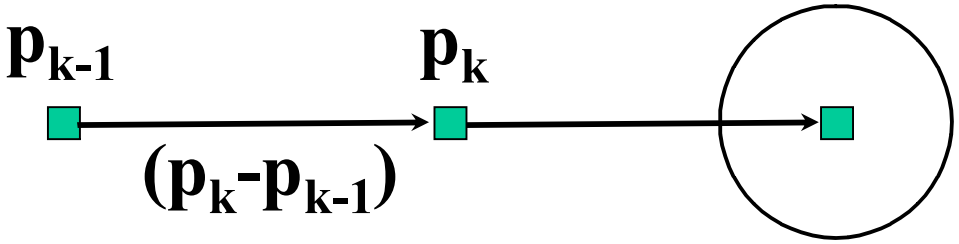
Constant position + bound on maximum interframe motion





constant position
prediction

Three-frame constant velocity prediction

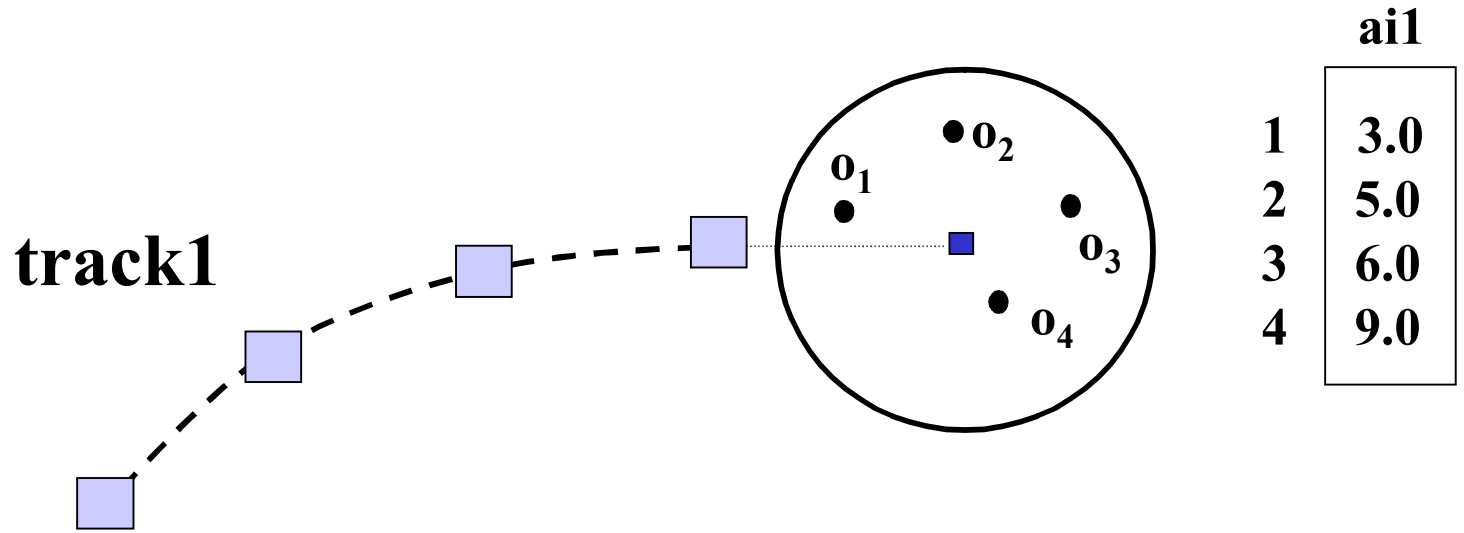


prediction
$p_k + (p_k-p_{k-1})$

typically, gating
region can be smaller

Global Nearest Neighbor (GNN)

Evaluate each observation in track gating region.
Choose "best" one to incorporate into track.

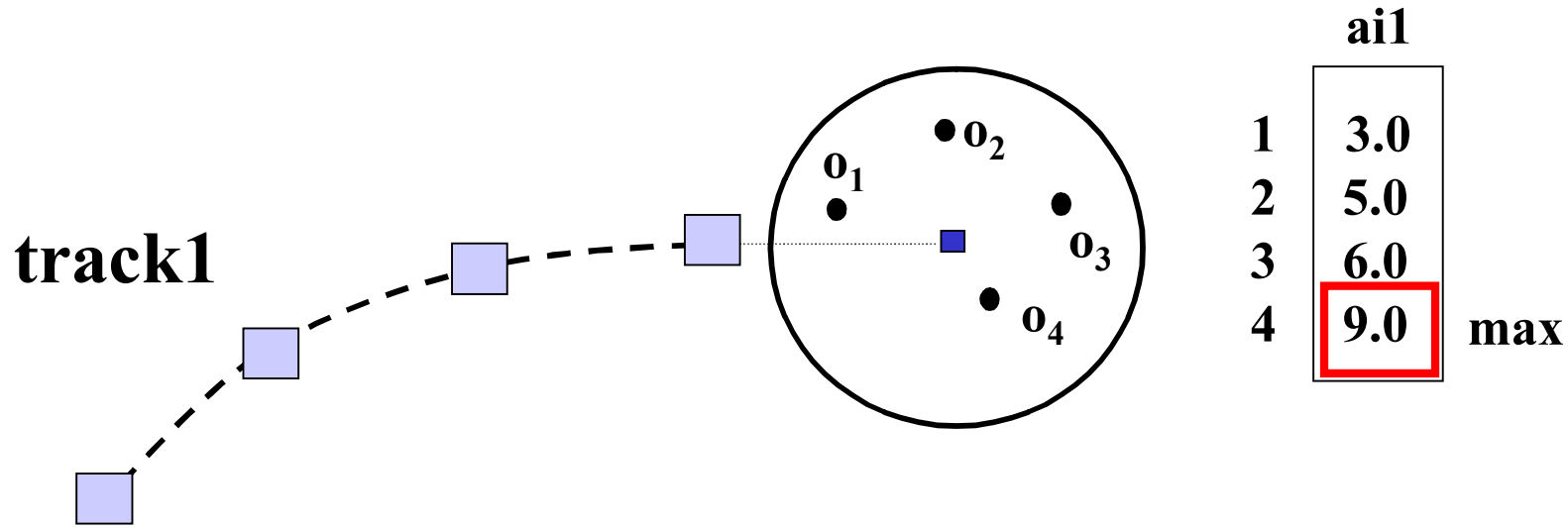


a_{1j} = score for matching observation j to track 1

Could be based on Euclidean or Mahalanobis distance to predicted location (e.g. $\exp\{-d^2\}$). Could be based on similarity of appearance (e.g. appearance template correlation score)

Global Nearest Neighbor (GNN)

Evaluate each observation in track gating region.
Choose "best" one to incorporate into track.



a_{i1} = score for matching observation i to track 1

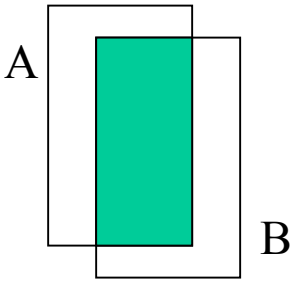
Choose best match $a_{m1} = \max\{a_{11}, a_{21}, a_{31}, a_{41}\}$

Data Association Scores

Similarity or affinity scores for determining the correspondence of blobs across frames is based on feature similarity between blobs.

Commonly used features: location , size / shape, velocity, appearance

For example: location, size and shape similarity can be measured based on bounding box overlap:



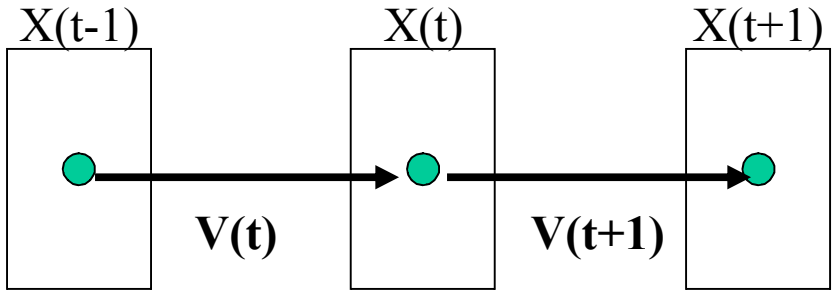
$$\text{score} = \frac{2 * \text{area}(A \text{ and } B)}{\text{area}(A) + \text{area}(B)}$$

A = bounding box at time t
B = bounding box at time t+1

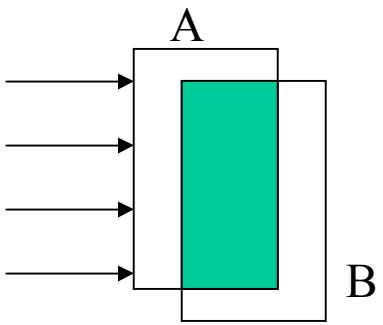
Robert Collins
CSE598C

Data Association Scores

It is common to assume that objects move with constant velocity



constant velocity
assumes V(t) = V(t+1)

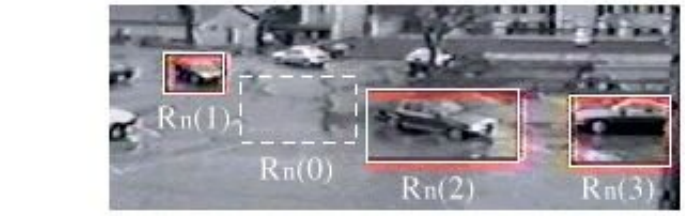


$$\text{score} = \frac{2 * \text{area}(A \text{ and } B)}{\text{area}(A) + \text{area}(B)}$$

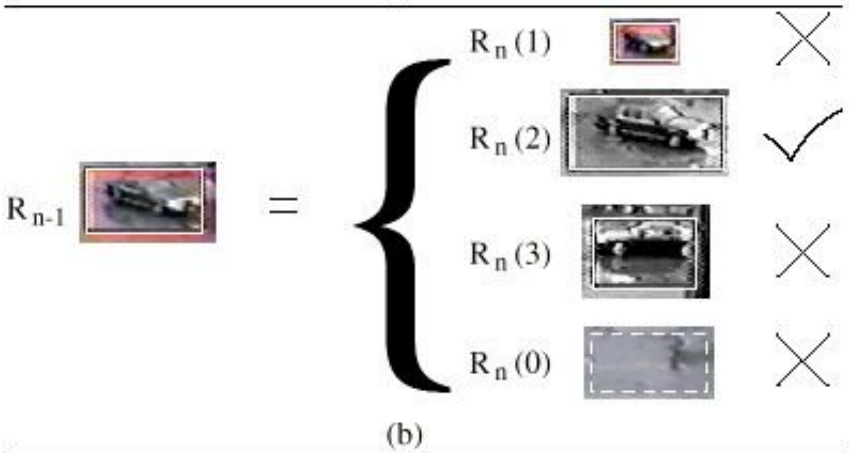
A = bounding box at time t, adjusted by velocity V(t)
B = bounding box at time t+1

Using Appearance Scores

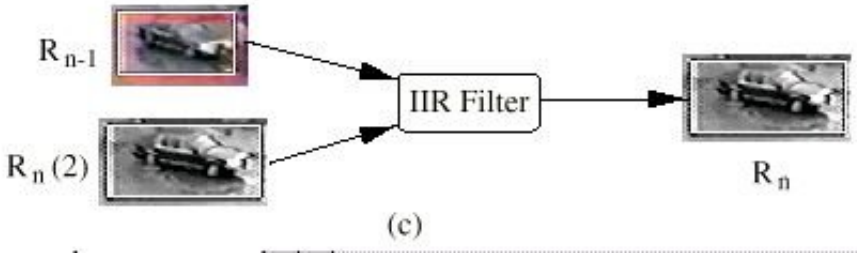
Correlation of image templates is an obvious choice (between frames)



(a)



(b)



(c)

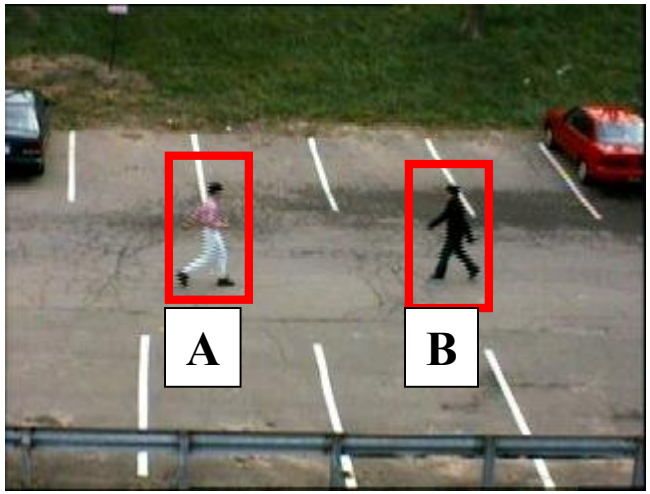
Extract motion blobs

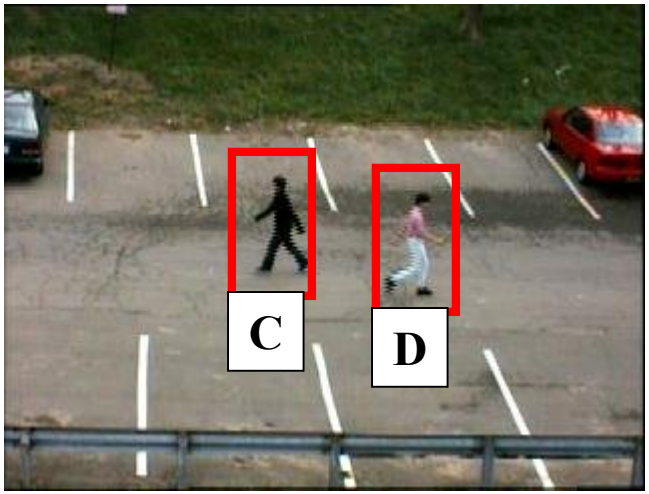
For object in previous frame, compute correlation score with all blobs in current frame. Pick one with highest score (suboptimal strategy).

Update appearance template of blobs

However, cross correlation is computationally expensive.

Example of Data Association After Merge and Split





$\Delta(A,C) = 0.39$
$\Delta(A,D) = 2.03$ ●

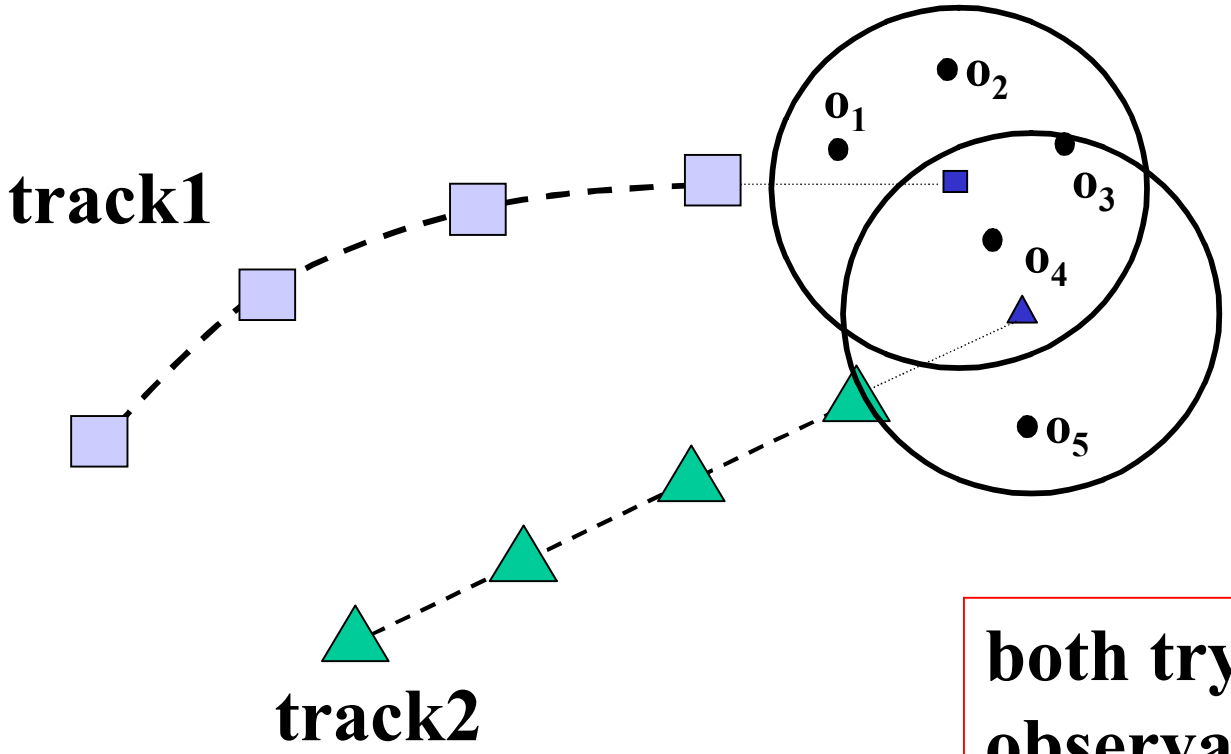
A -> D

$\Delta(B,C) = 2.00$ ●
$\Delta(B,D) = 0.23$

B -> C

Global Nearest Neighbor (GNN)

Problem: if do independently for each track, could end up with contention for the same observations.



	ai1	ai2
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

both try to claim observation o_4

Robert Collins
CSE598C

Linear Assignment Problem

We have N objects in previous frame and M objects in current frame. We can build a table of match scores m(i,j) for i=1...N and j=1...M. For now, assume M=N.

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

problem: choose a 1-1 correspondence that maximizes sum of match scores.

Robert Collins
CSE598C

Assignment Problem

Mathematical definition. Given an NxN array of benefits $\{X_{ai}\}$, determine an NxN permutation matrix M_{ai} that maximizes the total score:

maximize:

$$E = \sum_{a=1}^{N} \sum_{i=1}^{N} M_{ai} X_{ai}$$

subject to:

$$\forall i \ \sum_{a=1}^{A} M_{ai} = 1$$

$$\forall a \ \sum_{i=1}^{I} M_{ai} = 1$$

$$M_{ai} \in \{0, 1\}$$

constraints that say
M is a permutation matrix

The permutation matrix ensures that we can only choose one number from each row and from each column. (like assigning one worker to each job)

Example:

5x5 matrix of match scores

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

working from left to right, choose one number from each column, making sure you don't choose a number from a row that already has a number chosen in it.

How many ways can we do this?

$$5 \times 4 \times 3 \times 2 \times 1 = 120 \quad \text{(N factorial)}$$

Examples

0.95	0.76	0.62	0.41	0.06
0.23	**0.46**	0.79	0.94	0.35
0.61	0.02	**0.92**	0.92	0.81
0.49	0.82	0.74	**0.41**	0.01
0.89	0.44	0.18	0.89	**0.14**

score: 2.88

0.95	0.76	0.62	**0.41**	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	**0.92**	0.92	0.81
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	0.89	**0.14**

score: 2.52

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 4.14

Robert Collins
CSE598C

A Greedy Strategy

Choose largest value and mark it
For i = 1 to N-1
  Choose next largest remaining value that isn't in a row/col already marked
End

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	**0.92**	0.92	0.81
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	0.89	**0.14**

score: 3.77

not as good as our current best guess!

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 4.14

Is this the best we can do?

Solution Methods

maximize: $$E = \sum_{a=1}^{N} \sum_{i=1}^{N} M_{ai} X_{ai}$$

subject to: $$\forall i \ \sum_{a=1}^{A} M_{ai} = 1$$

$$\forall a \ \sum_{i=1}^{I} M_{ai} = 1$$

$$M_{ai} \in \{0, 1\}$$

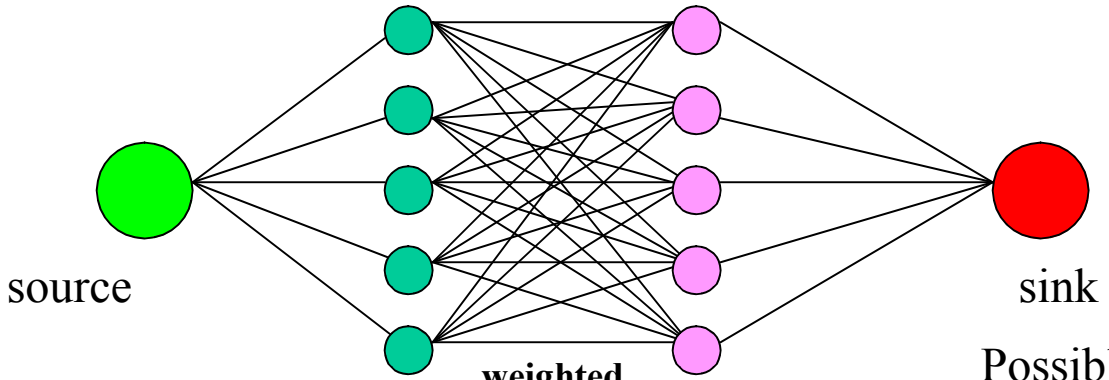
This has the form of a 0-1 integer linear program. Solution methods include branch and bound; cutting plane; etc. Bad (exponential) worst-case complexity... binary integer programming is NP-hard.

Max-Flow?

maximize: $$E = \sum_{a=1}^{N} \sum_{i=1}^{N} M_{ai} X_{ai}$$

subject to:
$$\forall i \ \sum_{a=1}^{A} M_{ai} = 1$$
$$\forall a \ \sum_{i=1}^{I} M_{ai} = 1$$
$$M_{ai} \in \{0, 1\}$$

The problem can also be viewed as a weighted bipartite graph, with nodes being row/col indices and edges being weighted by the matrix entries Xai. Perhaps this can be solved by mincut/maxflow?



Possible solution methods:
Ford-Fulkerson algorithm?

Assignment Problem

However, it is not a general max-flow problem, due to the constraints

maximize: $$E = \sum_{a=1}^{N} \sum_{i=1}^{N} M_{ai} X_{ai}$$

subject to:

$$\forall i \ \sum_{a=1}^{A} M_{ai} = 1$$
$$\forall a \ \sum_{i=1}^{I} M_{ai} = 1$$
$$M_{ai} \in \{0, 1\}$$

} constraints that say M is a permutation matrix

The permutation matrix ensures that we can only choose one number from each row and from each column. Translation into graph-speak: only one internal edge per node can be turned on. The solution must be a "matching". We want the maximally-weighted matching. ➔ Assignment problem!

Robert Collins
CSE598C

Hungarian Algorithm

Hungarian algorithm

From Wikipedia, the free encyclopedia

The **Hungarian algorithm** is a combinatorial optimization algorithm which solves assignment problems in polynomial time ($O(n^3)$). The first version, known as the **Hungarian method**, was invented and published by Harold Kuhn in 1955. This was revised by James Munkres in 1957, and has been known since as the **Hungarian algorithm**, the **Munkres assignment algorithm**, or the **Kuhn-Munkres algorithm**. In 2006, it was discovered that Carl Gustav Jacobi had solved the assignment problem in the early 19th century, and published posthumously in 1890 in the Latin language.[1]

The algorithm developed by Kuhn was largely based on the earlier works of two Hungarian mathematicians: Dénes König and Jenő Egerváry. The great advantage of Kuhn's method is that it is strongly polynomial (see Computational complexity theory for details). The main innovation of the algorithm was to combine two separate parts in Egerváry's proof into one.

hence the name

Robert Collins
CSE598C

Hungarian Algorithm

5x5 matrix of match scores

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

the version we will describe finds min-cost matchings. So we will subtract all our match scores from a large number (1.0) in this example, to turn them into costs.

5	24	38	59	94
77	54	21	6	65
39	98	8	8	19
51	18	26	59	99
11	56	82	11	86

Robert Collins
CSE598C

Hungarian Algorithm

5	24	38	59	94
77	54	21	6	65
39	98	8	8	19
51	18	26	59	99
11	56	82	11	86

step 1: subtract the minimal cost from each row

0	19	33	54	89
71	48	15	0	59
31	90	0	0	11
33	0	8	41	81
0	45	71	0	75

check if we are done: can we form a permutation matrix out of elements that have value 0 in this array? No... then continue.

Hungarian Algorithm

0	19	33	54	89
71	48	15	0	59
31	90	0	0	11
33	0	8	41	81
0	45	71	0	75

step 2: subtract the minimal cost from each col

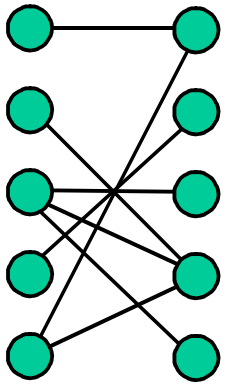
0	19	33	54	78
71	48	15	0	48
31	90	0	0	0
33	0	8	41	70
0	45	71	0	64

check if we are done: can we form a permutation matrix out of elements that have value 0 in this array? No... then continue.

Robert Collins
CSE598C

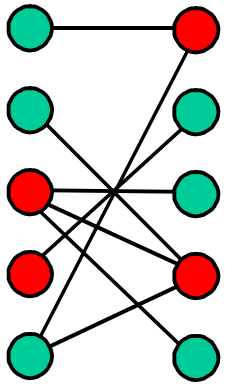
Hungarian Algorithm

0	19	33	54	78
71	48	15	0	48
31	90	0	0	0
33	0	8	41	70
0	45	71	0	64



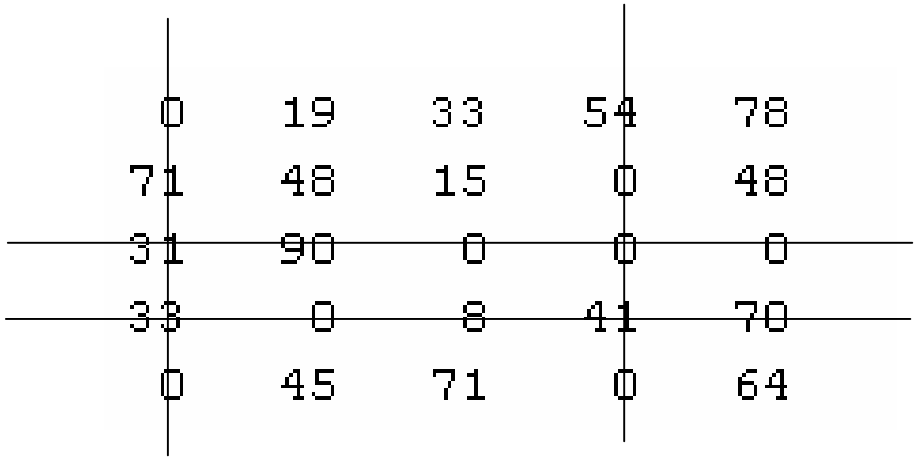
step 3: Draw as few row, col lines as possible to cover all the zeros. Note, in the graph that you are implicitly working with, this is a minimum vertex cover of the subgraph formed by zero-weight edges.

0	19	33	54	78
71	48	15	0	48
31	90	0	0	0
33	0	8	41	70
0	45	71	0	64



Robert Collins
CSE598C

Hungarian Algorithm



step 4: From the elements that are left, find the lowest value. Subtract this from all elements that are not struck. Add this to elements that are present at the intersection of two lines. Leave other elements unchanged.

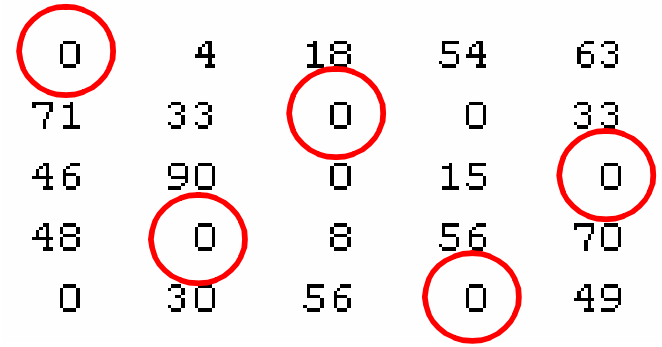
0	4	18	54	63
71	33	0	0	33
46	90	0	15	0
48	0	8	56	70
0	30	56	0	49

Robert Collins
CSE598C

Hungarian Algorithm

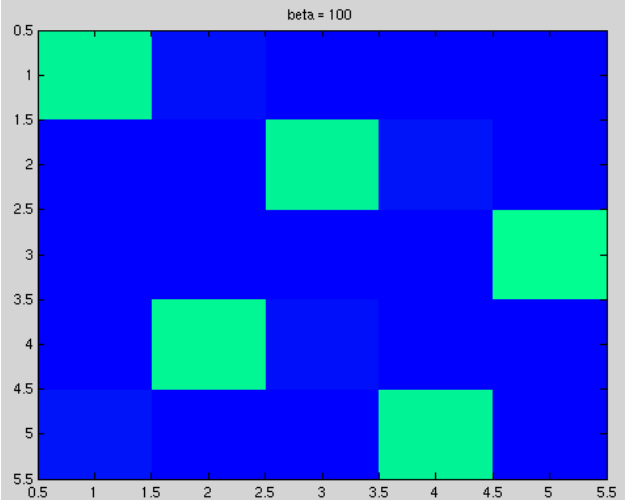
0	4	18	54	63
71	33	0	0	33
46	90	0	15	0
48	0	8	56	70
0	30	56	0	49

step 5: Check if done... is a max matching possible? If not, go back to step 3.



we can now form a permutation matrix with the zero elements (i.e. form a maximal matching in subgraph formed by zero-weight edges).
WE ARE DONE!

Result of Hungarian Algorithm



permutation matrix computed
by Hungarian Algorithm

0.95	0.76	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

score: 4.26

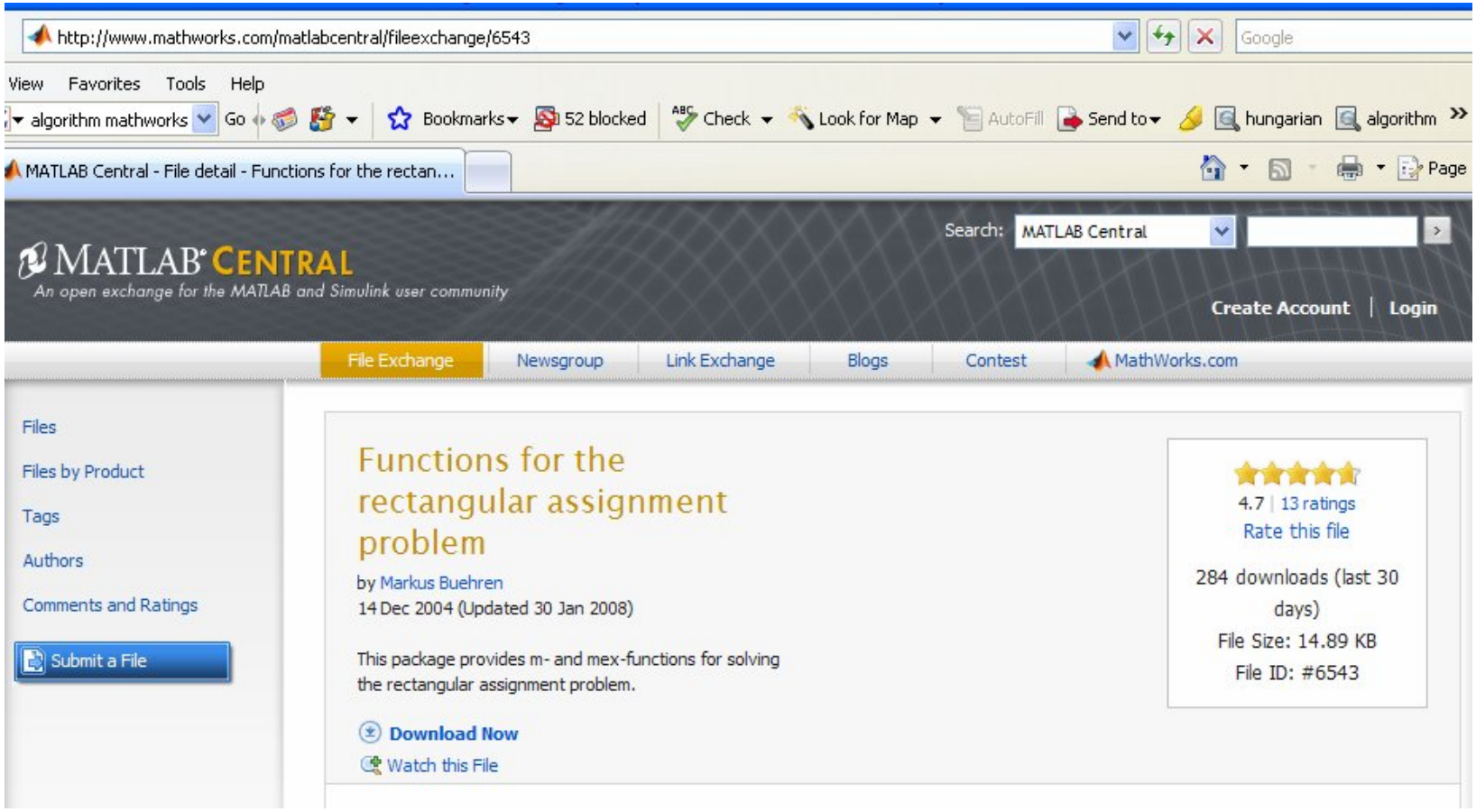
In this example, we can exhaustively search all 120 assignments.
The global maximum is indeed 4.26

Robert Collins
CSE598C

Handling Missing Matches

Typically, there will be a different number of tracks than observations. Some observations may not match any track. Some tracks may not have any observations.

That's OK. Most implementations of Hungarian Algorithm allow you to use a rectangular matrix, rather than a square matrix. See for example:

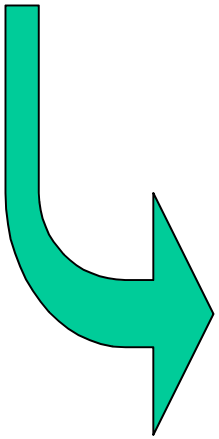


If Square Matrix is Required...

	track1	track2	
1	3.0	0	
2	5.0	0	
3	6.0	1.0	5x3
4	9.0	8.0	
5	0	3.0	

pad with array of small random numbers to get a square score matrix.

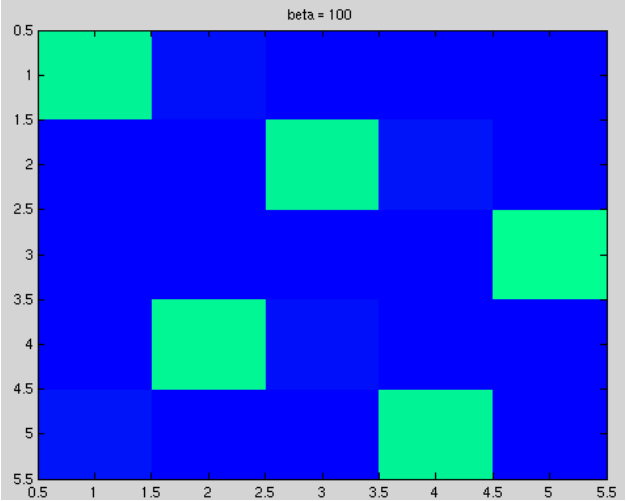
Square-matrix assignment



	track1	track2	
1	0	0	
2	0	0	5x3
3	1	0	
4	0	1	ignore whatever happens in here
5	0	0	

Robert Collins
CSE598C

K-Best Assignment



permutation matrix
computed by SoftAssign

0.95	0.76	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

score: 4.26

So far we know how to find the best assignment (max sum scores). But what if we also want to know the second best? Or maybe the top 10 best assignments?

Murty's K-Best Assignments

General Idea.

Start with best assignment.

Start methodically "tweaking" it by toggling matches in and out of the assignment

Maintain a sorted list of best assignments so far

During each iterative "sweep", toggle the matches in the next best assignment

The K best assignments are found in decreasing order, one per sweep

Robert Collins
CSE598C

1st sweep

0.95	0.76	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

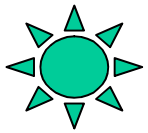
solution: (1,1)(4,2),(2,3),(5,4),(3,5)
constraints: none

constraints

~(1,1)

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(1,2),(4,3),(2,4),(3,5)
score 4.14



(1,1),~(4,2)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	**0.44**	0.18	0.89	0.14

(1,1)(5,2),(4,3),(2,4),(3,5)
score 3.88

(1,1)(4,2),~(2,3)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	**0.35**
0.61	0.02	**0.92**	0.92	0.81
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

(1,1)(4,2),(3,3),(5,4),(2,5)
score 3.93

(1,1)(4,2),(2,3),~(5,4)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	**0.92**	0.81
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	0.89	**0.14**

(1,1)(4,2),(2,3),(3,4),(5,5)
score 3.62

second best solution

Robert Collins
CSE598C

2nd sweep

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	0.89	0.14

solution: (5,1)(1,2),(4,3),(2,4),(3,5)
constraints: ~(1,1)

constraints

~(1,1),~(5,1)

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

(4,1)(1,2),(2,3),(5,4),(3,5)
score 3.74

~(1,1),(5,1),~(1,2)

0.95	0.76	**0.62**	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(4,2),(1,3),(2,4),(3,5)
score 4.08

~(1,1)(5,1),(1,2),~(4,3)

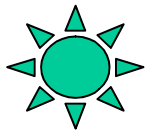
0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	0.74	**0.41**	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(1,2),(2,3),(4,4),(3,5)
score 3.66

~(1,1)(5,1),(1,2),(4,3),~(2,4)

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	0.79	0.94	**0.35**
0.61	0.02	0.92	**0.92**	0.81
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(1,2),(4,3),(3,4),(2,5)
score 3.66



third best solution

Robert Collins
CSE598C

1st scan, different order

0.95	0.76	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

solution: (1,1)(4,2),(2,3),(5,4),(3,5)
constraints: none

constraints

~(3,5)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	**0.35**
0.61	0.02	**0.92**	0.92	0.81
0.49	**0.82**	0.74	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

(1,1)(4,2),(3,3),(5,4),(2,5)
score 3.93

(3,5),~(5,4)

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	0.79	**0.94**	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(1,2),(4,3),(2,4),(3,5)
score 4.14

(3,5)(5,4),~(2,3)

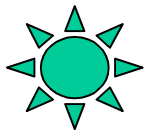
0.95	0.76	0.62	0.41	0.06
0.23	**0.46**	0.79	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	**0.74**	0.41	0.01
0.89	0.44	0.18	**0.89**	0.14

(1,1)(2,2),(4,3),(5,4),(3,5)
score 3.85

(3,5)(5,4),(2,3),~(4,2)

0.95	**0.76**	0.62	0.41	0.06
0.23	0.46	**0.79**	0.94	0.35
0.61	0.02	0.92	0.92	**0.81**
0.49	0.82	0.74	**0.41**	0.01
0.89	0.44	0.18	0.89	0.14

(4,1)(1,2),(2,3),(5,4),(3,5)
score 3.74



second best solution is found again.