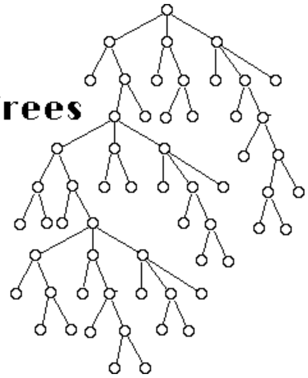
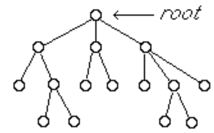
This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dennis-bricker@uiowa.edu

Search Trees



author

Search Trees



- Each node of the **search tree** for a problem represents a **subset of feasible solutions** of the problem
- The **root** of the tree represents the set of all feasible solutions of the problem
- The **descendents** of each node of the tree represent a **partition** of the set represented by that node

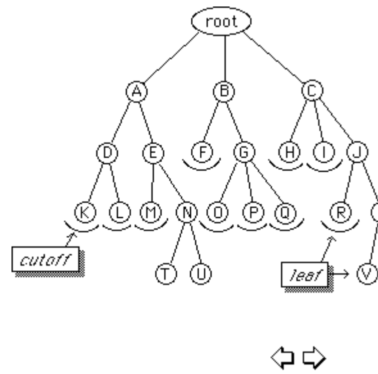
A collection of subsets B_i of set A (i=1,2,...t) is a **partition** if

$$B_1 \cup B_2 \cup B_3 \cdots \cup B_t = A$$

and

$$B_i \cap B_j = \varnothing \quad \text{if } i \neq j$$

Search Tree



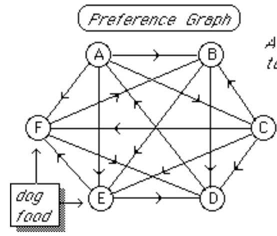
Example: Ranking Nodes in a Preference Graph

In many experiments (especially in the social sciences, when numerical measurement of attributes are difficult or impossible), one is required to **rank** a set of objects by comparing only **two at a time**.

Example

Six different dog foods are to be ranked according to their appeal to dogs.
Each day, 2 of the 6 are served to a dog, who indicates his preference by finishing it first.

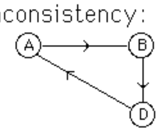
Preference Graph



Preference Matrix

	A	B	C	D	E	F
A	-	1	1	0	1	1
B	0	-	0	1	1	0
C	0	1	-	1	1	1
D	1	0	0	-	0	0
E	0	0	0	1	-	1
F	0	1	0	1	0	-

In the dog food example, the dog exhibited some inconsistency: for example,



he preferred A over B,
B over D,
and D over A!

How can we establish a "good" ranking?

Methods for Ranking

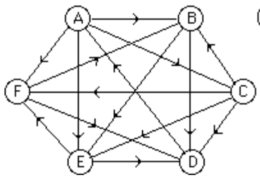
- ranking by score: the score of an object is the number of pairs in which it is preferred (i.e., the row-sum of the preference matrix).
  — ties may occur
  — assumes every possible pair was compared

	A	B	C	D	E	F	***score***
A	-	1	1	0	1	1	**4**
B	0	-	0	1	1	0	**2**
C	0	1	-	1	1	1	**4**
D	1	0	0	-	0	0	**1**
E	0	0	0	1	-	1	**2**
F	0	1	0	1	0	-	**2**

For example,
A > C > B > E > F > D
or C > A > F > E > B > D
etc.

Methods for Ranking

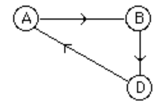
- **ranking by Hamiltonian path:** find a path through every node of the preference graph such that each node is preferred over its successor.
  For example, A→C→B→E→F→D
  or A→C→E→F→B→D



(several such paths may exist!)

Methods for Ranking

- **ranking with minimum discrepancies**
  A discrepancy is an instance in which X is ranked above Y, but Y is preferred to X



For example, the ranking A > B > D has one discrepancy (i.e., A>D)

— does not assume that every pair was compared!
— is a difficult problem to solve

Using a Search Tree for Minimum Discrepancy Ranking

Two different methods for partitioning:

- choose a pair of objects X & Y which have not been ranked.
  Form two subsets of rankings:
  --those in which X > Y, i.e., X is ranked above Y
  --those in which Y > X, i.e., Y is ranked above X

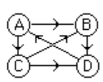
Second method of partitioning:

- an object is assigned to a position in the ranking

e.g., in the first partition, **n** nodes are created, in each of which one of the **n** objects is assigned to the **first** position in the ranking, and

in the second partition, **n−1** nodes are created, one for each of the remaining **n−1** objects which might be assigned to the **second** position in the ranking, etc.

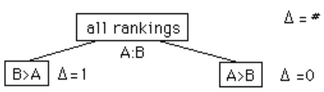
Example



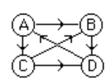
	A	B	C	D	score
A	-	1	1	0	2
B	0	-	0	1	1
C	0	1	-	1	2
D	1	0	0	-	1

First Partitioning Method

Δ = # discrepancies

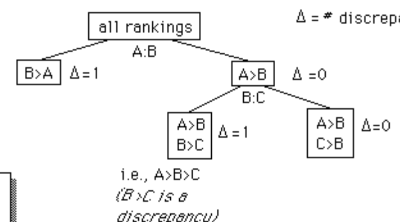


We will partition the most promising node, that with no discrepancies

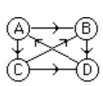


	A	B	C	D
A	-	1	1	0
B	0	-	0	1
C	0	1	-	1
D	1	0	0	-

Δ = # discrepancies

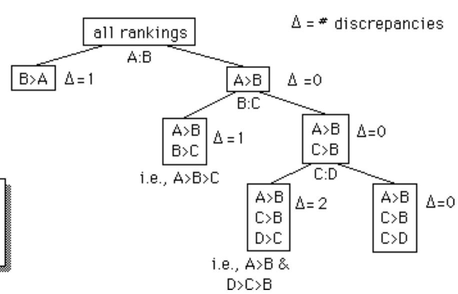


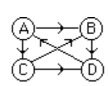
Again, we partition the most promising node



	A	B	C	D
A	-	1	1	0
B	0	-	0	1
C	0	1	-	1
D	1	0	0	-

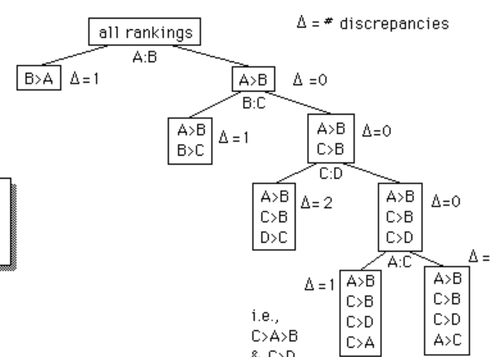
Δ = # discrepancies

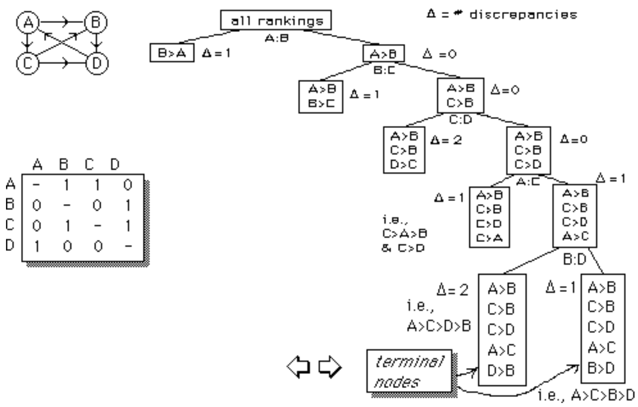


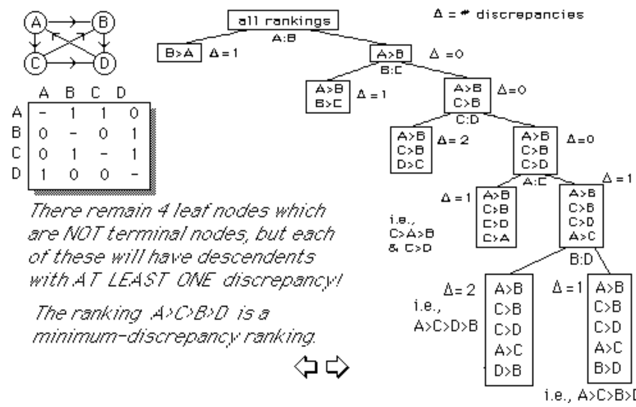


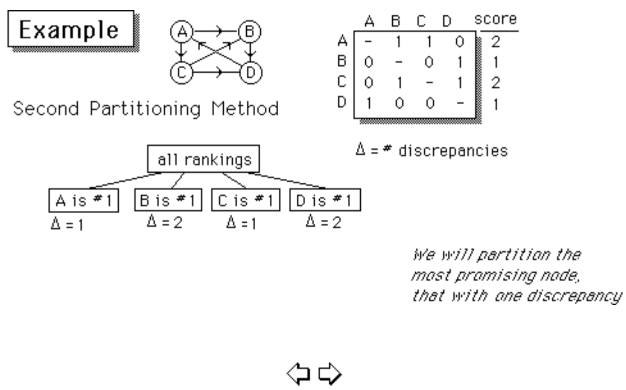
	A	B	C	D
A	-	1	1	0
B	0	-	0	1
C	0	1	-	1
D	1	0	0	-

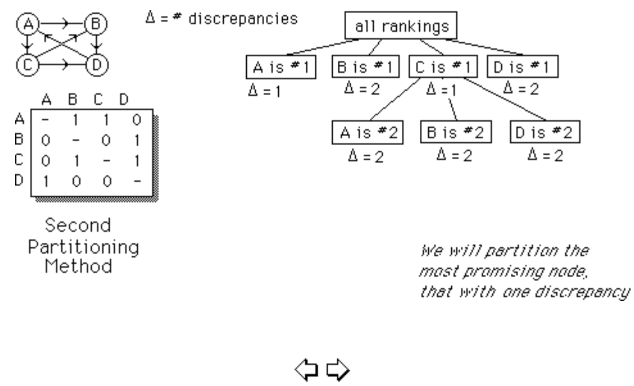
Δ = # discrepancies

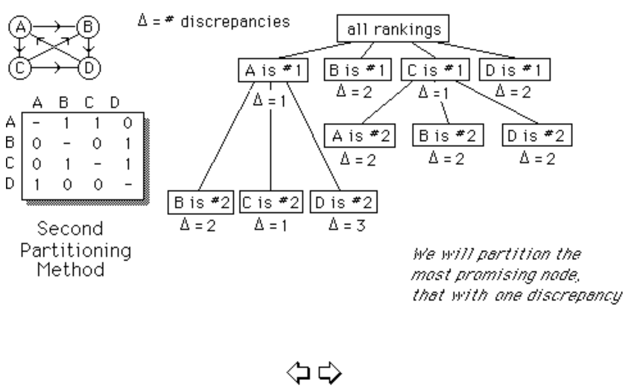


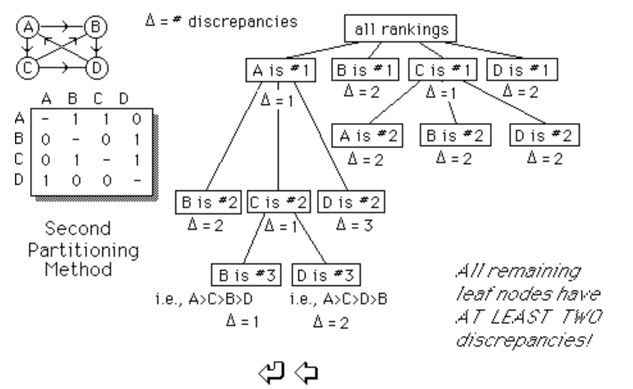

Δ = # discrepancies
all rankings
A:B
B>A Δ = 1
A>B Δ = 0
B:C
A>B B>C Δ = 1
A>B C>B Δ = 0
C:D
A>B C>B D>C Δ = 2
A>B C>B C>D Δ = 0
A:C
Δ = 1 A>B C>B C>D C>A
i.e., C>A>B & C>D
Δ = 1 A>B C>B C>D A>C
B:D
Δ = 2 A>B C>B C>D A>C D>B
i.e., A>C>D>B
Δ = 1 A>B C>B C>D A>C B>D
i.e., A>C>B>D
terminal nodes


Δ = # discrepancies
all rankings
There remain 4 leaf nodes which are NOT terminal nodes, but each of these will have descendents with AT LEAST ONE discrepancy!
The ranking A>C>B>D is a minimum-discrepancy ranking.
i.e., A>C>D>B
i.e., A>C>B>D


Example
Second Partitioning Method
score
Δ = # discrepancies
all rankings
A is #1 Δ = 1
B is #1 Δ = 2
C is #1 Δ = 1
D is #1 Δ = 2
We will partition the most promising node, that with one discrepancy


Δ = # discrepancies
all rankings
Second Partitioning Method
A is #1 Δ = 1
B is #1 Δ = 2
C is #1 Δ = 1
D is #1 Δ = 2
A is #2 Δ = 2
B is #2 Δ = 2
D is #2 Δ = 2
We will partition the most promising node, that with one discrepancy


Δ = # discrepancies
all rankings
Second Partitioning Method
A is #1 Δ = 1
B is #2 Δ = 2
C is #2 Δ = 1
D is #2 Δ = 3
We will partition the most promising node, that with one discrepancy


Δ = # discrepancies
all rankings
Second Partitioning Method
B is #3 i.e., A>C>B>D Δ = 1
D is #3 i.e., A>C>D>B Δ = 2
All remaining leaf nodes have AT LEAST TWO discrepancies!