Introduction to Stat 232B

Statistical Computing and Inference in Vision and Image Science

This graduate level course introduces a broad range of advanced ***algorithms*** for statistical inference and learning on graph structures. These algorithms could be used in vision, pattern recognition, speech, bio-informatics, data mining. We organize these algorithms in three methods according to the underlying representation.

1. ***Descriptive methods***: algorithms working on various graphs where the nodes/vertices represent states of the same semantic level. E.g, constraint-satisfaction, relaxation-labeling, dynamic programming, Viterbi, belief propagation, Gibbs sampler, Sequential Monte Carlo, Swendsen-Wang

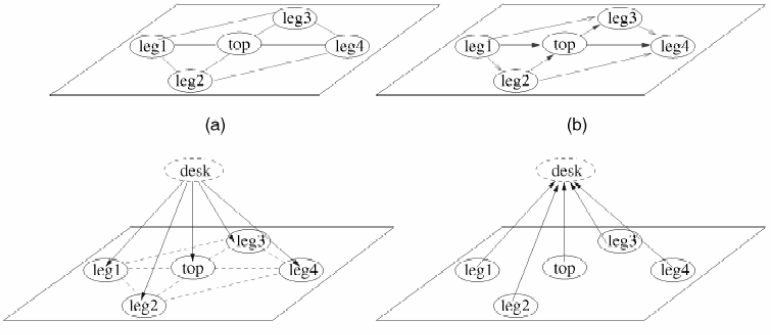
2. ***Generative methods:*** algorithms working on hierarchic graph representations where one level of vertices semantically generate the nodes at the level below as parts/components. E.g. search on And-Or graphs, heuristic search, parsing, Metropolis-Hastings, Markov chain Monte Carlo, reversible jumps.

3. ***Discriminative methods*:** algorithm working on selecting features for discriminating various classes. taught in stat231, e.g. clustering, decision tree, boosting, SVM.

To Clarify the terminology

Descriptive or declarative
(Constraint-satisfaction, Markov random fields, Gibbs, Julesz ensemble)

Variants of Descriptive
(Causal Markov Models, Markov chain, Markov tree, DAG etc)



Generative (+ Descriptive)
(hidden Markov, hierarchic model decomposing whole to parts)

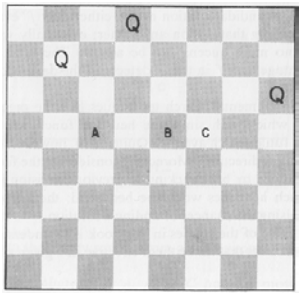
Discriminative
(discriminating the whole using the parts)

Outline

1. Some classic inference problems in CS and statistics
2. State space and representation
3. Objectives of these inference tasks
4. Optimization Criterion

Ex 1: 8-Queen problem

Put 8 queens in a 8x8 board so that they are safe: i.e. no two queens occupy the same row, column, or diagonal lines.



Inference 1: 8-Queen problem

This is a *constraint-satisfaction* problem on a 8x8 grid.

Let's define s be a solution, s could be a binary 8x8 matrix or a list of the coordinates for the 8 queens.

Define the solution in a set:

$$\Omega* = \{ s : \quad h_i(s)=1, \quad i=1,2,\ldots,46 \}$$

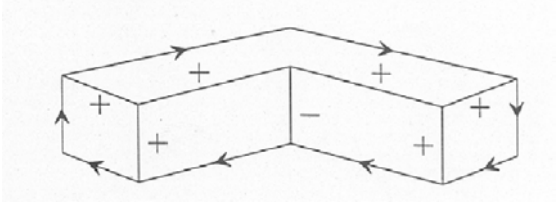
$h_i(s)$ is a hard (logic) constraints respectively for the 8 row, 8 column, 30 diagonal lines.

The computational problem is

$$\boxed{find \qquad s \in \Omega^*}$$

Ex 2: Line drawing interpretation

Label the edges of a line drawing (graph) so that they are consistent



This is also *constraint-satisfaction* problem on a graph G=<V,E>.

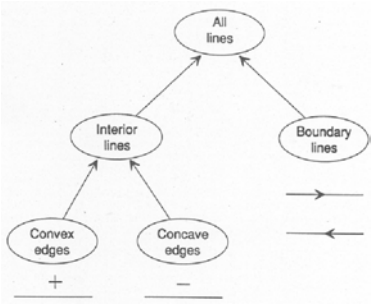
Define the solution in a set:

$$\Omega* = \{ s : \quad h_i(s)=1, \quad i=1,2,\ldots,|V| \}$$

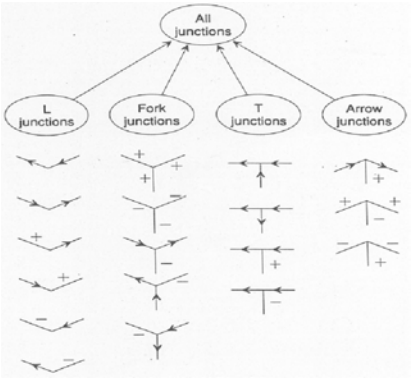
$h_i(s)$ is a hard (logic) constraints respectively for consistence at each vertex.

Ex 2: Line drawing interpretation

allowed edge labels

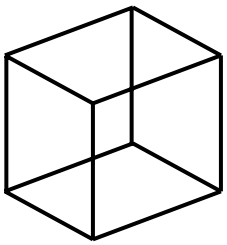


allowed junction labels

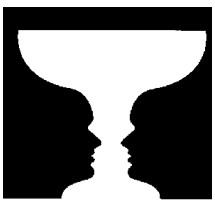


Example of multiple Solutions in Visual Inference

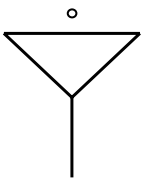
Like the 8-queen and line drawing problems, visual scenes can have a set of solutions as well. A real world image may have a huge set of solutions and they are assigned a "score" or "weight".



Nicker cube



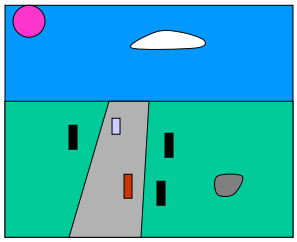
Vase vs. face



bikini vs. martini

A common property is that the individual elements are strongly coupled and those strongly coupled elements must change their labels together.

Ex 3: Scene interpretation



Interpret an image with a set object labels

{ sun, cloud, sky, field, road, house, car, …}

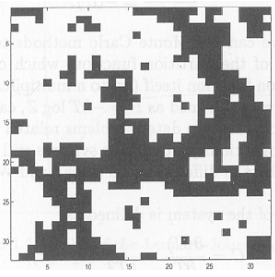
The set of constraints can be hard and soft
(soft means probabilistic here)

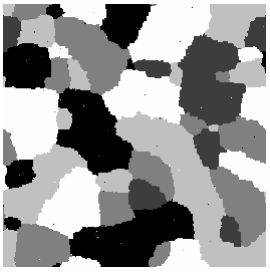
$$\Omega^* = \{ (s, p(s)) : \; E_p(h_i(s)) = a_i , \quad i = 1,2,...,M \; \}$$

$$s \sim \Omega^*, \quad or \quad s \sim p(s), \; or \quad s^* = \arg\max p(s)$$

Ex 4: simulating Gibbs models

Simulating the Ising/Potts models, Mumford/Shah models
("*Simulation*" extends "*relaxation*", "*Gibbs model*" extends "*Constraints*")





This such high dimensional space, the concept of a "solution" is extended to the *typical configurations*, the set of typical configurations is often huge !

Ex 6: Simulating typical protein structures

This such high dimensional space, the concept of a "solution" is extended to the *typical configurations*, the set of typical configurations is often huge !

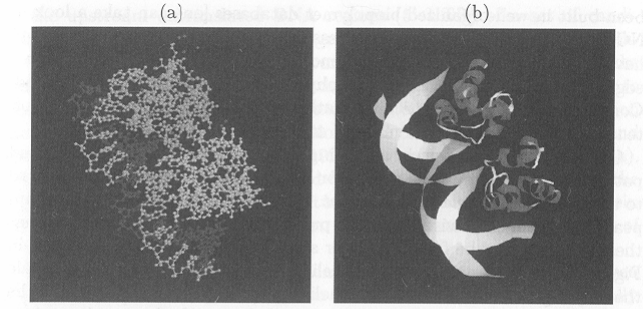


FIGURE 1.4. (a) A ball-and-stick plot of the interaction between a regulatory protein in yeast, 3CRO, and the DNA segment to which it binds. (b) The same structure as in (a), but expressed by a ribbon representation widely used in the protein structure modeling community. [From ref book by Liu]

More examples

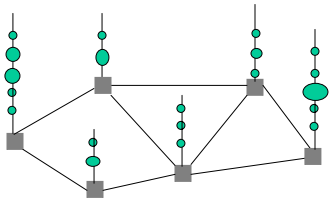
There are many more similar examples, e.g.

image restoration, image segmentation, graph partition/coloring, shape from stereo/motion/shading …

Common properties:

1. A graph representation G=<V, E>.
   G could be directed, undirected, such as chain, tree, DAG, lattice, etc.

2. hard constraints or soft "energy" preference between adjacent vertices.



Descriptive methods

These problems belong to the descriptive family. The computing algorithms includes:

Relaxation-Labeling, Dynamic programming (I consider HMM as descriptive model not generative),
Belief propagation,
Gibbs sampler, Swendsen-Wang.

Issues in algorithm design:

1. Visiting scheme design and message passing.
   which step is more informative, relax more constraints (like line-drawing). In general, the ordering of Gibbs kernels
2. Computing joint solution or marginal belief.
   the marginal believe may be conflicting to each other.
3. Clustering strongly-coupled sub-graphs for effective moves.
   the Swendson-Wang ideas.
4. Computing multiple solutions.

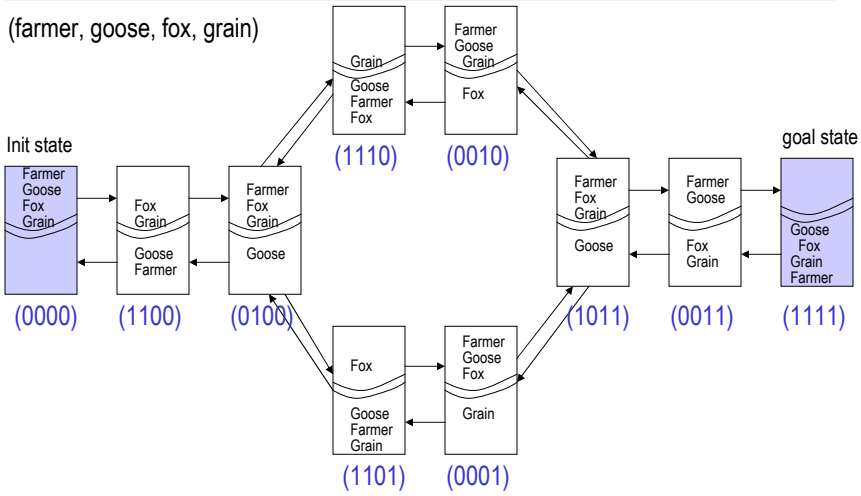
Ex 7: farmer, goose, fox, and grain

A *farmer* wants to move himself, a silver *fox*, a fat *goose*, and some Tasty *grain* across a river. Unfortunately, his *boat* is so tiny he can Take only one of his possessions across on any trip. Worse yet, an Unattended fox will eat a goose, and an unattended goose will eat Grain.

How can he cross the river without losing his possessions?

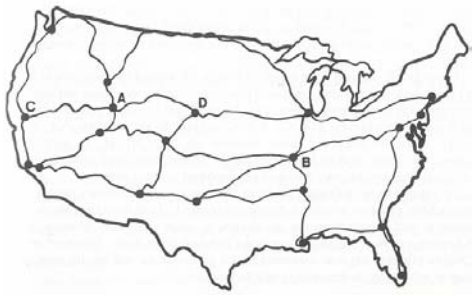
This can be formulated as finding a path in the state-space graph (next page).
In the coin example, the path is further extended to and-or graph.

The State Space Graph



Ex 8: Traveling salesman problem

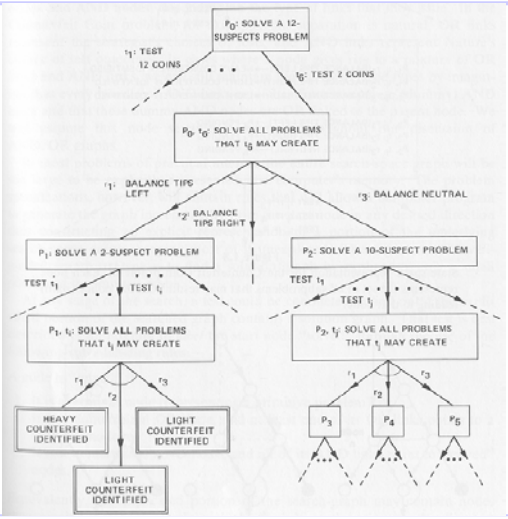
This is another typical search problem for finding a path with is a loop.
It also relax the hard constraint to a soft cost.



Ex 9: 12 Counterfeit coin problem

Given 12 coins, one is known to be heavier or lighter than the others. Find that coin with no more than 3 tests using a two-pan scale.

This generates the And-Or graph representation.



And-Or Graph is also called "hyper-graph"

The and-Or graph represents the decomposition of task into sub-tasks recursively.

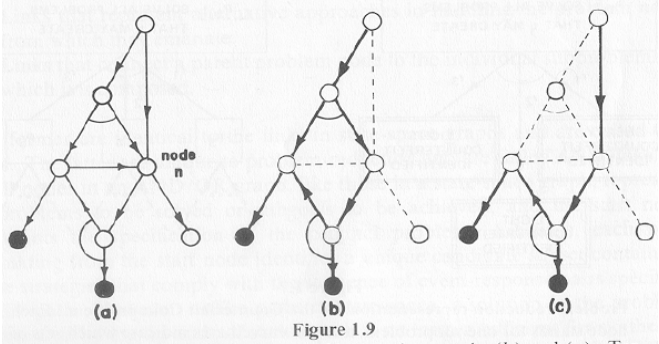
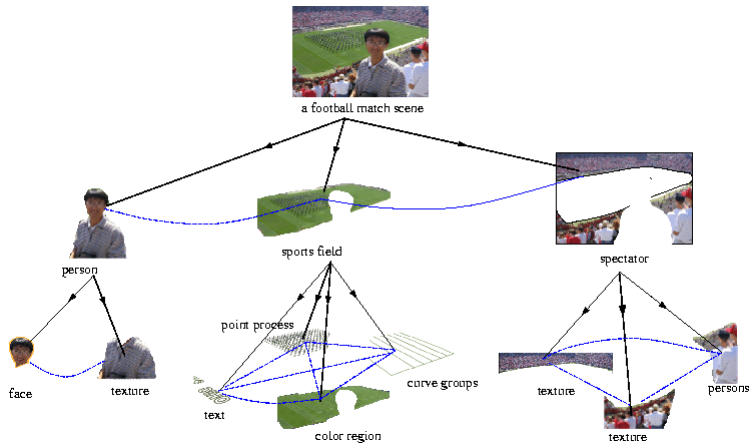


Figure 1.9
An AND/OR graph (a) and two of its solution graphs (b) and (c). Terminal nodes are marked as black dots.

Ex 10: Images parsing

Parsing an image into its constituent visual patterns. The parsing graph below is a solution graph with AND-nodes



Characterization of these problems

The search problem has the following components

1. An initial state
2. A number of "*operators*" to *generate* a set of new states (children / off-springs),
3. The Or-nodes for alternative ways
4. The And-nodes for sub-tasks that must be solved in combination
5. Metrics for each operator
6. Leaf nodes are solvable directly
7. The final solution is an AND_OR graph ---called solution graph.

Motivation for Integration

It turns out that we need to integrate all three methods for complex inference tasks for example, image parsing in computer vision.

1. Construct the parsing graph by generative methods.
   (Heursitic search in AI, Markov chain Monte Carlo, reversible jumps)

2. Passing messages in a parsing graph by descriptive methods.
   (relaxation labeling, belief propagation, dynamic programming, MCMC simulation)

3. Making hypotheses and proposals by discriminative methods.
   (clustering, adaboosting, edge detection, RANSAC etc.)

To see the connections between these methods, we need to trace literature in computer science, statistics, computer vision, language understanding etc. Make connections between them. Then pool a global picture.

Various Criteria for Algorithm Design

For simple problems, like 8-queen, we may seek exact solutions, but for very complex problems, like TSP, the exact solution needs a lift-time search, and we have to search for nearly optimal solutions.

Let *A* be an algorithm that take external input I, and I follows a probability $f(I)$.

For example, I could be an input image, an adversary chess player, a city map in a TSP problem etc. $f(I)$ characterizes the *ensemble* of problems.

Definition 1. *A* is *optimal* if it can always find an exact solution s for any I.

$$s \in \Omega^*(I) \qquad \forall I$$

Various Criteria for Algorithm Design

Definition 2. *A* is *near optimal* if it can always find a solution *s* within ε-distance to the exact solution for any I. e.g. we often only care about near optimal for TSP.

$$s \in \Omega^*_\varepsilon(\mathrm{I}) \qquad \forall \mathrm{I}$$

Definition 3. *A* is *approximate optimal* if it can probably find a solution *s* within ε-distance to the exact solution for any I.

$$p(s \in \Omega^*_\varepsilon(\mathrm{I})) > 1-\delta \qquad \forall \mathrm{I}$$

Definition 4. *A* is *approximate optimal for ensemble f(I)* if it can probably find a solution *s* within ε-distance to the exact solution for the ensemble. This relaxes the worst case to average cases.

$$p(s \in \Omega^*_\varepsilon(\mathrm{I})) > 1-\delta$$

Stochasticity in Algorithm

When the problem becomes complex, we have to relax the optimality criteria and the Probability in the relaxed criteria arise in three aspects

1. *Probabilistic decisions in the algorithm.*
   e.g. the Gibbs sampler, the Metropolis-Hastings steps. The algorithm does not move deterministically.
2. *The ensemble probability.*
   e.g. we should focus the most likely input rather than the worst cases.
3. *The heuristics using in algorithm design*
   e.g. heuristics are often rule of thumb, common senses, they are not fully reliable, but "work" *most of the time*.

Heuristics

A poem quoted in Pearl 84.

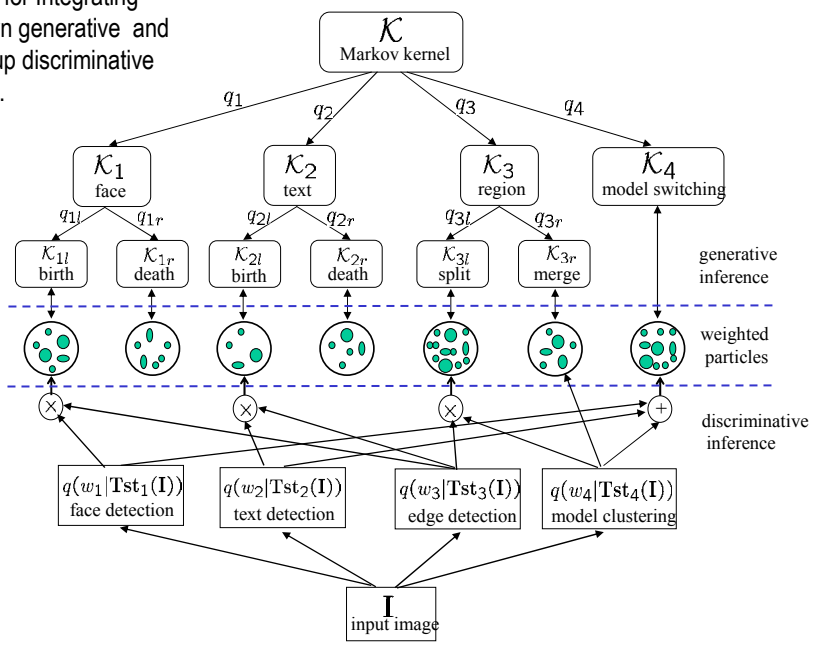
> "Heuristics, Patient rules of thumb,
> So often scorned: Sloppy! Dumb!
> Yet, Slowly, common sense come" (ODE to AI)

In the 1980s, the probabilistic models are not well known to the CS search literature, Pearl viewed heuristics as ways

"inform the search algorithm with simplified models"

In our understanding today, the heuristics are discriminative methods which inform or drive the Markov chain search.

Diagram for Integrating Top-down generative and Bottom-up discriminative Methods.



MCMC as a common framework

To summarize, we have several types of problems, and MCMC is a common framework

1. *Simulation, i.e. draw fair (typical) samples from a pdf.*

$$s \sim p(s), \ s \text{ is a configuration.}$$

2. *Integration/computing in very high dimensions, i.e. to compute*

$$c = \int p(s) f(s) ds$$

3. *Optimization*

$$s^* = \arg\max p(s)$$

4. *Learning with hidden variables*

Many Open Questions

1. The necessity of MCMC inference
   Under what condition does a stochastic algorithm beat a deterministic one?
2. Integrating the heuristic search mechanisms in MCMC design.
   MCMC only remember one past state, while AI search opens many solutions.
3. Optimal control strategy, and mechanisms for constructing the solution graphs.
4. Developing a mathematical foundation for the use of heuristics, or we may have to reformulate heuristics.
5. Balancing computational power with costs for each types of moves.
6. The integration of three types of methods: descriptive, generative, discriminative.

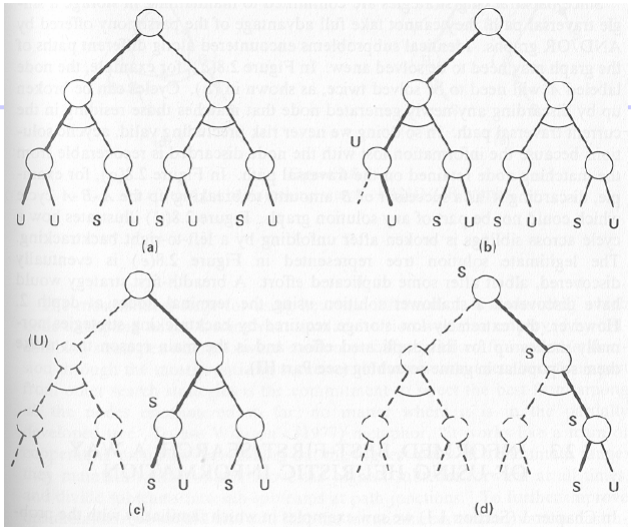


Figure 2.7
Typical steps in the execution of backtracking search of an AND/OR tree. The heavy line represents the traversal path, whereas the broken lines represent portions of the tree that can be purged from memory.

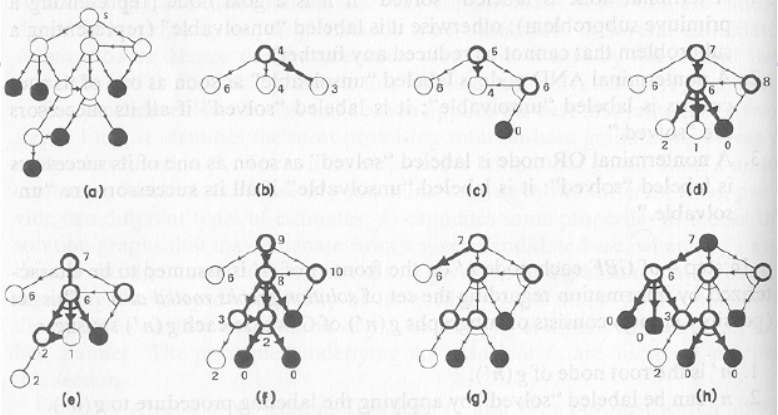
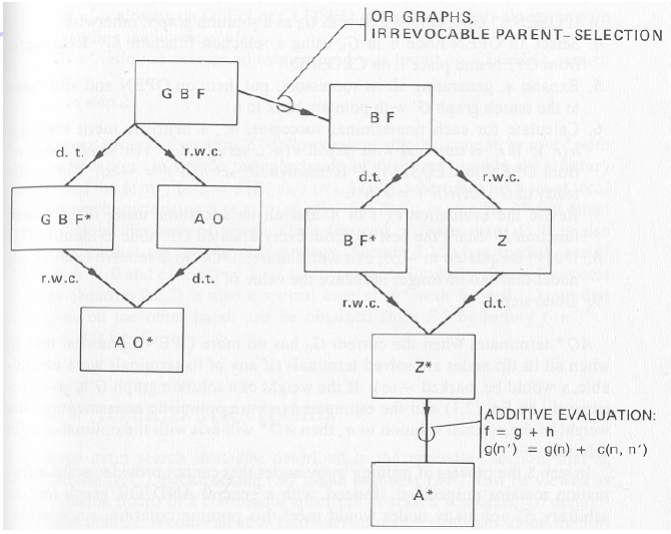


Figure 2.9
Successive steps in the execution of general-best-first (GBF) search on the implicit AND/OR graph of part (*a*). Solid circles represent solved nodes, heavy hollow circles nodes in CLOSED, and thin circles nodes in OPEN. The heavy lines stand, at each stage, for the current most promising solution base.


OR GRAPHS,
IRREVOCABLE PARENT- SELECTION
G B F
B F
d. t.
r.w.c.
d.t.
r.w.c.
G B F*
A O
B F*
Z
r.w.c.
d.t.
r.w.c.
d.t.
A O*
Z*
ADDITIVE EVALUATION:
f = g + h
g(n') = g(n) + c(n, n')
A*