Genetic Algorithms

21/7/15

→ ~~this~~ Consider a set of solutions at a particular time. (this is where it differs from hill climbing & SA)

In GA,
we start with an initial set of solns. Then iteratively ~~select~~ emulate natural genetic ~~processes~~ process on this set (selection)

genetic operators → i) crossover
ii) mutation

population → set of solns
fitness value → evaluation func.

GA

⇒GA → adaptive, changes with the changing environment

⇒Only some processes can be emulated.

⇒ GA operates on encoded version of solutions.

Eg $f(x)$ ⇒ maximize this objective fn. $x \in [0, 31]$
GA acts on encoding of 5, not directly on 5.

⇓

binary encoding

⇒ Use obj. fn. $f(x)$ [acts as environment]

GA used when : Why GA?

→ search space is large

~~fn. should be conti~~ ← known not to be perfectly smooth
not unimodal / not well understood

~~Why GA?~~

→ Fitness fn. is noisy

→ Search time should be min.

When we can reduce the time & get a near optimal soln., we use GA.

Works well with multimodal search space (more than one optimal solns)

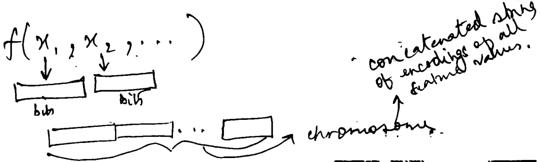
⇒ Solution encoded using l bits → 2^l solutions

⇒ K iterations

M solutions / iteration

complexity : $M \times K$

We use GA when $\boxed{M \times K \ll 2^l}$



Depending on range of parameter, the no. of bits read. for encoding is determined

→ A chromosome is a coded possible soln.

→ We generally keep the chromosome size fixed

$$x_j = \text{Lower bound} + \frac{\sum_{i=0}^{\#bits-1} bit_i * 2^i}{2^{\#bits}-1} * (\text{Upper bound} - \text{Lower bound})$$

→ high fitness value to better ~~solns~~ solns.

No. of copies propagated to next gen ∝ fitness value

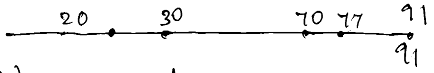
Roulette Wheel

Area of segment on wheel assigned to chromosome proportional to ~~the~~ fitness value.

Eg Fitness Values → 20, 10, 40, 7, 14

total fitness = 91

prob. of worst chromosome getting selected $= \frac{7}{91}$



We may not get best chromosome here if env. is competitive in nature, so we go for linear normalization selection,

$f \longrightarrow$ we create this fn based on problem

$\downarrow$

obj. fn. (maximize or minimize)

fitness fn is always maximized.

fitness fn. = g(obj. fn. · f)

If fitness value increases with increase in value of f, fitness fn = $g(f(x))$

else

fitness fn = $g\left(\frac{1}{f(x)}\right)$

or

$g(-f(x))$

Linear Normalization Celection

chromosomes are closely fitted ie, have fitness values close to each other.

$$\begin{tikzcd} 90.2 \arrow[d] & 90.1 \arrow[d] & 89.9 \arrow[d] & 89.8 \arrow[d] & \rightarrow \text{actual fitness} \\ 100 \arrow[r, leftrightarrow] & 90 \arrow[r, leftrightarrow] & 80 \arrow[r, leftrightarrow] & 70 & \rightarrow \text{modified fitness} \end{tikzcd}$$

same decrement parameter (d) used, hence linear.

if d is small, it again becomes a close competitive environment.

No. of chromosomes copied to next gen

$$\rightarrow C_i = \frac{n \cdot f_i}{\sum f_i} = \frac{f_i}{\left(\frac{\sum f_i}{n}\right)} = \frac{f_i}{\bar{f}}$$

round off to some integer by some scheme

In probabilistic, use modified fitness value and use random no. to select. In probabilistic, a bad chromosome may get selected.

In deterministic, use modified fitness value & use ~~deterministic~~ above formula. Here, a bad chromosome won't be selected.

disadv.: a bad chromosome never gets selected; as generation increases, similarity betw. strings increases until all strings become same but it may not be the optimal soln. This is premature convergence.

In selection, old solns get copied, no new soln created.

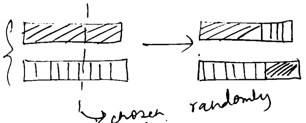
We use genetic operators to get new solns:

Normally prob. of crossover is kept very high (0.8 - 0.9)

Afterwards mutation may occur but it occurs with very low prob. (bec this is a sudden alternation; if prob is high, half of the bits change frequently & it becomes a random search)

Parent : C1

Parent : C7



in the same iterations, these parents will not be considered again after cross-over.

⇓

this is basically swapping of segments & hence exchange of information.

~~In case~~

We use multipoint crossover bec.

Then we allow values of multiple features to change simultaneously but in single pt. crossover many features will retain same value. Only some may ~~[illegible]~~ change.

On newly generated crossover chromosome, apply mutation on it bit by bit using Pm (mutation prob.)

Eg

1010
1000
10....
10....

Why mutation?

i) 2nd bit is ~~1~~ 0 for all chromosomes in initial set. In this case, ~~also~~ crossover will not make 2nd bit 1 ever.

⇓

For this, mutation is needed.

ii) We also need mutation to bring about a drastic change.

get this child 0011

parents (1) 1100
(2) ?

Eg

110 | 001
011 | 011 } parent

110011
011001 } children

what features of parents do we see in children?

Crossover → does ~~exploration~~ exploitation ; → current search space exhausted & moved one to diff area

Mutation → does exploration } combined in GA

Replacement Strategy

Generational → replace all parents with all children

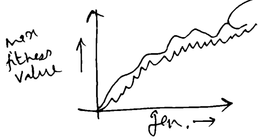
Steady State →

Termination

→ fixed no. of iter

→ String structures no similar

Elitism



→ a very good soln. ~~may be~~ is lost.

In elitist model we ~~store~~ keep track of the best solns. for every generation.

In EGA, the max. value of f either remains same or increases

GA → allows parallelism

coded parameter → accuracy may be increased

Next day → predicate analysis

Predicate Calculus

23/9/15

Say initially we have a set of English sentences.

Derive info. from it (new sentence) following certain rules.

Earlier, we had a 3×3 int matrix for 8 puzzle problem & some seq. operators.

Now we have a database of English sentences

Inference Rules

Represent Eng. sent. in some form (some other language) & the simplest of these languages is Propositional logic. An extension of this First Order PL. An extension of that again is Second Order PL.

Nilsson → book on AI

We call this System

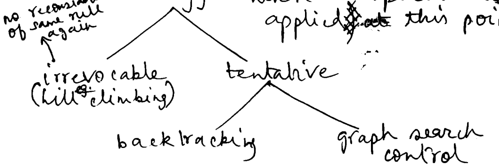
AI production System

has 3 distinct components

→ how ~~prob~~ data needs to be stored

→ Operation (set of operators)

→ Control Strategy (which path will be chosen when an operator is applied ~~at~~ this point)


no reconsideration of same rule again
irrevocable (hill climbing)
tentative
backtracking
graph search control

statement nos.

Procedure Production

1) DATA ~~Data~~ ← initial database

2) Until ~~do~~ DATA satisfies termination condition do

```
3) begin
4)     Select some rule r in the set of rules
       that can be applied to ~~data~~ DATA
5)     DATA ← Result of applying r to DATA
6) end
```

4 different symbols used in PL:

Var
const
fn
predicate → designates some relation present on the sentence

A well formed formula(wff) ~~for~~ → a combination of these symbols ~~to~~ form a legitimate expression.

Eg

i) Nilson wrote Principles of AI.

↓

Write in predicate calculus.

$$\text{WROTE}\,(\overset{\text{const.}}{\text{Nilsson}},\ \overset{\text{const.}}{\text{Principles of AI}}) \rightarrow$$

value of this exprn. is either T or F

WROTE ↓ reln./predicate

ii) The house is yellow.

~~Color~~

~~Color (House, yellow)~~

'the' house → a definite house, so const. ~~const.~~

better to write H_1 / House1 since const.

Color(H_1, yellow)

Value(Color, H_1, yellow)

iii) John's mother is married to John's father.

Married(Father(John), Mother(John))

$\downarrow$

predicate

Diff. betwn. pred. & fn.

pred → ret. true or f

fn. → ret. a value

We need connectives to connect predicates:

$\lor$, $\land$, $\rightarrow$

$\downarrow$ or, $\downarrow$ and, $\downarrow$ implication

Some times we need negation: ¬/~

eg

iv) John lives in a yellow house.

Lives(John, $House_1$) $\land$ Color($House_1$, Yellow)

v) John plays chess or badminton.

Plays(John, Chess) $\lor$ Plays(John, Badminton)

· if A then B
$A \rightarrow B$

vi) If the car belongs to John, then it is green.

Belongs (John, car_1) → Color (Car_1, Green

Upto this we were using PL.

Now we move on to FOPL: → here we use Variable → So, we need quantifier

vii) All elephants are grey.

~~Color(elephant, gr grey)~~

~~Animal~~

We need some representation of "for all".

i) existential
ii) universal

~~∀elephant, Color(elephant, grey)~~

$(\forall x)[Elephant(x) \rightarrow Color(x, grey)]$

or
$Animal(x) \wedge Elephant(x)$

viii) There is a person who wrote Geeta.

$(\exists x)[Person(x) \wedge Wrote(x, Geeta)]$

In (vii) & (viii), x is a bound variable.

If we use quantification in case of fn. or pred., then it is SOPL

De Morgan
Distributive } all valid.
Associative

$$\sim(\forall x) \equiv (\exists x)$$
$$\sim(\exists x) \equiv (\forall x)$$

(ix)

For every set X, there is a set Y such that cardinality of Y is greater than that of X.

$$(\forall x)\left[Set(x) \wedge (\exists y)\left[Set(y) \wedge Greater(Card(x), Card(y))\right]\right]$$

~~$(\forall x)[Set(x) \wedge (\exists y)[Set(y) \wedge Greater(|x|, |y|)]]$~~

$$(\forall x)[Set(x) \wedge (\exists y)[Set(y) \wedge (\exists a)[Card(x,a) \wedge (\exists b)[Card(y,b) \wedge Greater(b,a)]]]]$$

$$(\forall x)\left[Set(x) \longrightarrow (\exists y)\left[Set(y) \wedge \cancel{Card}(\exists a)\left[Card(\cancel{a}x,a) \wedge (\exists b)\left[Card(y,b) \wedge greater(b,a)\right]\right]\right]\right]$$

$\cancel{(\forall x)\exists}$

$\Downarrow$ neater form

$$(\forall x)\left\{Set(x) \longrightarrow (\exists y)(\exists a)(\exists b)\left[Set(y) \wedge Card(x,a) \wedge Card(y,b) \wedge greater(b,a)\right]\right\}$$

Note where we use AND
and where we use "implies"

$A \to B$ mean $\sim A \vee B$

24/9/15

Different types of Rules of Inference → can only be applied to some wffs

1) Modus Ponems
produce W_2 from W_1 and $W_1 \Rightarrow W_2$

2) Universal ~~sp~~ specialization
produce $W(A)$ from $(\forall x)W(x)$

~~3~~ Combined, produce $W_2(A)$ from
$(\forall x)[W_1(x) \Rightarrow W_2(x)]$
and $W_1(A)$

$S = \{A/x\}$ → var x substituted by const. A

derived wff } called → theorem
proof

Unification

match ~~contain~~ certain sub exprns

Finding substitution of 'terms' for 'vars'.
↓
vars/const/fn

Eg

Substitution instance :– given $P[x, f(y), B] \leftarrow E_i$ g

we can have following 4 diff substitution instances:

$P[z, f(W), B]$ $\quad S_1 = \{ z/x, W/y \}$

$P[x, f(A), B]$ $\quad S_2 = \{ A/y \}$

$P[g(z), f(A), B]$ $\quad S_3 = \{ g(z)/x, A/y \}$

$P[C, f(A), B]$ $\quad S_4 = \{ C/x, A/y \}$

$S_1 \equiv$ alphabetic variant
↓
var. subst.ed by var.

$S_4 \equiv$ ground instance
var subed by const.

$$S = \{ t_1/v_1, t_2/v_2, \ldots, t_n/v_n \}$$

→ each occurrence of a var has same term substituted for it. [x subed by y ∀x]

→ No var can be expressed by a term containing the same var [x cannot be subed by f(x)]

→ Eg This is the way we write:

→ $P[z, f(w), B] = P[x, f(y), B]_{S_1}$ (Exprn.)(Subst.)

→ $S_1 S_2$

$$\{ \underbrace{g(x,y)}_{\text{term}} / \underbrace{z}_{\text{var}} \} \{ A/x, B/y, C/w, D/z \} ~~= \{ g(A,B)/z~~$$

~~$= \{ D/g(A,B), C/W \}$~~ (*) (PTO)

~~Eg 1: All men are mortal. Socrates is a man~~

(~~S_4 = {~~ Associative but not commutative

$(S_1 S_2) S_3 = S_1 (S_2 S_3)$

$S_1 S_2 \neq S_2 S_1$

Unifiable

$\{E_i\}$

$E_1 S = E_2 S = \dots$

a set of exprns $E = \{E_1, E_2, E_3\}$ is unifiabl if we can find a subst S s.t. $E_1 S = E_2 S = E_3 S$ ie we get a singleton set

$S \Rightarrow$ unifier of $\{E_i\}$

eg

$S = \{A/x, B/y\}$ We can have another sub: g = {B/y

$E = \{P[x, f(y), B], P[x, f(B), B]\}$

Yes, S is the unifier of ~~s~~ E.

mgu, g of E_i has property —

$\{E_i\} S$ → most general unifier

$$\{E_i\} S = \{E_i\} g \overset{S'}{S'}$$

~~{B/y} {~~

$$E_1 S = ~~S~~ E_1 \overset{g}{\{B/y\}} \overset{S'}{\{A/x\}} \quad E_2 S = E_2 \overset{g}{\{B/y\}} \overset{S'}{\{A/x\}}$$

---

eg of ① → All men are mortal.
Socrates is a man
⇓
Socrates is mortal.

---

from prev pg

(*) Substn obtained by applying S_2 to the terms of S_1

$g(A, B)/z$

& then adding any pairs of S_2 having

variables not occurring among the variables of S_1

$S_1 S_2 = \{ g(A,B)/z, \ A/x, B/y, C/W \}$

Sets of literals		Most general common substitution instances
$[P(x), P(A)]$	—	~~A/x~~ $P(A)$
$\{P[f(x), y, g(y)], P[f(x), z, g(x)]\}$	—	$P[f(x), x, g(x)]$
$\{P[f(x, g(A, y)), g(A, y)], P[f(x, z), z]\}$	—	$P[f(x, g(A,y)), g(A,y)]$

Unification Algorithm: (Iterative)   K ≡ iter no.

1) Set $K=0$ and $mgu_K = \{ \}$
2) if the set E_{mgu_K} is a singleton set, then
   stop;
   mgu_K is an mgu of E
   Find the disagreement set D_K of E_{mgu_K}.
3) if there is a var v and term t in D_K st. that v does not occur in t
   Put $mgu_{K+1} = mgu_K \{t/v\}$
   Set $K = K+1$ & return to Step 2

otherwise

Stop, E is not unifiable.

Eg $E = \{P(f(x,x),A), P(f(y,f(y,A)),A)\}$

$K = 0$

$mgu_0 = \{\}$

~~$Emgu_0 = \{P[x,f(y),B], P[x,f(B),B)]\}$~~

~~$D_{\cancel{k}0} = \{x,y\}$~~

$Emgu_0 = \{P(f(x,x),A), P(f(y,f(y,A)),A\}$

$D_{\cancel{k}0} = \{x,y\}$

$mgu_1 \overset{\cancel{m}}{=} \{y/x\}$

$K = 1$

~~$K = 1$~~

$Emgu_1 = \{P(f(y,y),A), P(f(y,f(y,A),A)\}$

D_1 ~~$= \{A/y\}$~~ $= \{f(y,A)/y\}$

~~$mgu_2 = \{y/x\}\{f(y,A)/y\}$~~

Cond. fails.

So stop.

[redacted]

28/9/15

Resolution → a type of rule of inference applied to certain class of wff's called clause.

a set of wff's where only disjunctions of literals are present

Represent info. in clause form by 9 steps.

$$(\forall x)\{P(x) \Rightarrow \{(\forall y)[P(y) \Rightarrow P(f(x,y))] \wedge \sim(\forall y)[Q(x,y) \Rightarrow P(y)]\}\}$$

Step 1

Eliminate implication symbol

$\boxed{A \Rightarrow B \text{ means } \sim A \vee B}$

$$(\forall x)\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge \sim (\forall y)[\sim Q(x,y) \vee P(y)]\}\}$$

Step 2 [Note $\sim(\forall y)f(y) \equiv (\exists y)\sim f(y)$]

Reduce scope of negation. (Use DeMorgan's Laws)

$$(\forall x)\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge (\exists y)[Q(x,y) \wedge \sim P(y)]\}\}$$

~~$(\exists y)[Q(x,y) \vee P(y)]\}\}$~~

Step 3

Standardize variables.

Use a single variable for a single quantifier

$$(\forall x)\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge (\exists z)[Q(x,z) \wedge \sim P(z)]\}\}$$

Step 4

Eliminate '$\exists$' (existential quantifier sign)

(i) $(\forall y)[(\exists x)P(x,y)]$

~~there~~ For all y, there exists x possibly depending on y s.t. P(x, y) is true, so x is a fn. of y.

$\downarrow$ becomes

$(\forall y)[P(f(y), y)]$

This method is called skolemization & f is termed as skolem fn.

~~$(\forall x)\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge$~~

(ii) if no $\forall$ ~~exists~~ exists before $\exists$, replace $\exists$ by skolem constant.

$$(\exists x)P(x) \rightarrow P(A)$$

$$(\forall x)\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge [Q(x, g(x)) \wedge \sim P(g(x))]\}\}$$

Step 5

Convert to ~~pre~~ Premex form.

Move all $\forall$'s at the front of the expression (called prefix of exprn) & the rest of the ~~[illegible]~~ exprn is called matrix.

$$(\forall x)(\forall y)\{\sim P(x) \vee \{[\sim P(y) \vee P(f(x,y))] \wedge [Q(x, g(x)) \wedge \sim P(g(x))]\}\}$$

Step 6

Put matrix in conjunctive normal form by repeatodly using distributive Law.

$$x_1 \overset{\text{disjunction}}{\vee} (x_2 \overset{\text{conjunction}}{\wedge} x_3) \rightarrow (x_1 \vee x_2) \wedge (x_1 \vee x_3)$$

[conjunction of disjunctions]

$$x)(\forall y)\Big\{\Big(\sim P(x) \vee \big[\sim P(y) \cdot \vee P(f(x,y))\big]\Big) \wedge \Big(\sim P(x) \vee \big[Q(x,g(x)) \wedge \sim P(g(x))\big]\Big)\Big\}$$

$$> (\forall x)(\forall y)\Big\{\Big(\sim P(x) \vee \sim P(y) \vee P(f(x,y))\Big) \wedge \Big(\big(\sim P(x) \vee Q(x,g(x))\big) \wedge \big(\sim P(x) \vee \sim P(g(x))\big)\Big)\Big\}$$

Here, we have 3 clauses.

~~S~~Step 7

Eliminate $\forall$.

Step 8

Eliminate $\wedge$.

~~step~~ $x_1 \wedge x_2 \rightarrow \{x_1\}, \{x_2\}$

Step 9

Rename variables (single var appears only once in an expr.)

$\{\sim P(x) \vee \sim P(y) \vee P(f(x,y))\}, \{\sim$

Parent Clauses	Resolvent	Comments
$\{P\}, \{\sim P \vee Q\}$	Q	~~P & $P \to Q$~~ modus ponens
$\{P \vee Q\}, \{\sim P \vee Q\}$	Q	merge ~~the clause~~
$\{P \vee Q\}, \{\sim P \vee \sim Q\}$	$(Q \vee \sim Q) \wedge (P \vee \sim P)$	two possible resolvents, both are tautologies (exprn. is always true)
$\{\sim P \wedge P\}$	NIL	empty clause sign of contradiction
$\{\sim P \vee Q\}, \{\sim Q \vee R\}$	$\sim P \vee R$	chaining ($P \to Q$, $Q \to R$ so $P \to R$)

Resolution Refutation System

S → given set of expressions □

try to prove

W → goal wff

ie. we create $S \cup \sim W$

if $S \cup \sim W$ ---→ NIL

then S ---→ W

Production System for RRS

Let S be a set of clauses (base set)

```
~~Step1~~
~~1)~~ Clauses ← S
  until NIL is a member of Clauses do
  begin
      Select 2 distinct ~~dis~~ resolvable clauses,
       $C_i$ & $C_j$.
      Produce its resolvant $r_{ij}$.
      ~~Clauses ← $r_{ij}$~~
      Add $r_{ij}$ to Clauses.
  end
```

Eg

Whoever can read is literate.
Dolphins are not literate.
Some dolphins are intelligent.
With this info, prove that ~~some~~
Some who are intelligent cannot read

$$(\forall x)\{(\text{Read}(x)) \longrightarrow \text{Literate}(x)\}$$

~~$\sim(\forall x)\{D\,\text{Animal}(x,$~~

$(\forall y)\{$ ~~$D(y) \Rightarrow$~~ $\text{Dolphin}(y) \longrightarrow \sim\text{Literate}(y)\}$

$$(\exists z)\{\text{Dolphin}(z) \wedge \text{Intelligent}(z)\}$$

$$\downarrow$$

$$(\forall x)\{\sim\text{Read}(x) \vee \text{Literate}(x)\}$$

$$(\forall y)\{\sim\text{Dolphin}(y) \vee \sim\text{Literate}(y)\}$$

$$\text{Dolphin}(A) \wedge \text{Intelligent}(A)$$

~~$\{\sim\text{Read}(x) \vee \text{Literate}(x)\}, \{\sim\text{Dolphin}(y) \vee \sim\text{Literate}(y)\},$~~
~~$\{\text{Dolphin}(A)\}, \{\text{Intelligent}(A)\}$~~

$$\sim(\exists w)[\text{Intelligent}(w) \wedge \sim\text{Read}(w)]$$

negation of reqd. to prove

$$\downarrow$$

$$(\forall w)[\sim\text{Intelligent}(w) \vee \text{Read}(w)]$$

$S = \{ \{\sim Read(x) \vee Literate(x)\}$ ✓,

$\{\sim Dolphin(y) \vee \sim Literate(y)\}$,

$\{Dolphin(A)\}$, $\{Intelligent(A)\}$,

$\{\sim Intelligent(w) \vee Read(w)\}$

$\}$

~~$\sim I(w) \vee R(w)$~~

1) $\{\sim \overset{(A)}{I} \vee \overset{(A)}{R}\}$, $\{\sim R \vee L\} \rightarrow \sim I \vee L$

Add $\sim I \vee L$ to S

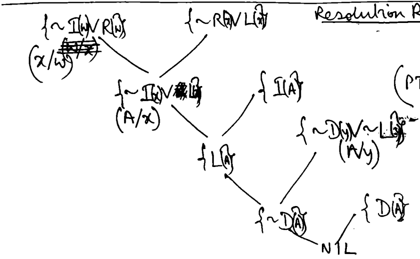
2) $\{I\}$, $\{\sim I \vee L\} \rightarrow \{L\}$

Add $\{L\}$ to S

3) $\{L\}$, $\{\sim D \vee \sim L\} \rightarrow \{\sim D\}$

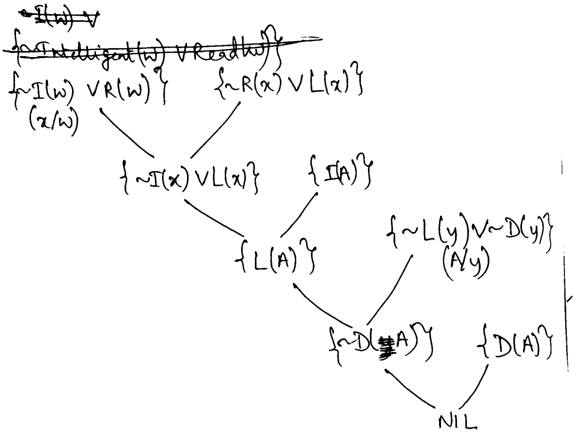
add $\{\sim D\}$ to S

4) $\{D\}$ $\{\sim D\} \longrightarrow$ NIL



Resolution Refutation Tree

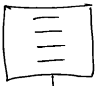
(PTO for clearer diag.)



~~6/~~
7/10/15

Production Systems for Resolution Refutation

S, a set of clauses (base set)



set of infor (given wff)

↓ convert to

clauses (used for resolution)

↓

resolution

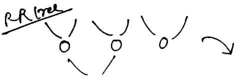
[redacted]

If we need to prove W (goal wff) from the given wffs, then append ~W to S ie $S \cup \sim W$
If $S \cup \sim W \rightarrow$ NIL,
W can be logically derived from S

~~Let S be the set of clauses:~~
~~Procedure Resolution~~
~~1) Clauses ← S~~
~~until NIL is a member~~

Control strategies for resolution refutation

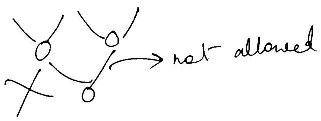
i) Breadth first strategy
Compute all 1st level resolvents, then 2nd level & so on.



It is complete but less efficient.
We can reach goal node (NIL).
Efficiency depends on depth of the tree

ii) Linear Input form strategy
Each resolvent has at least one parent

selonging to the base set.



Eg

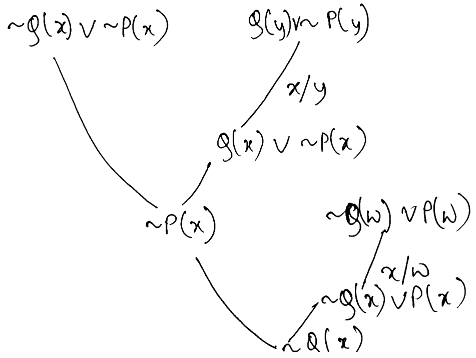
$Q(u) \vee P(A)$

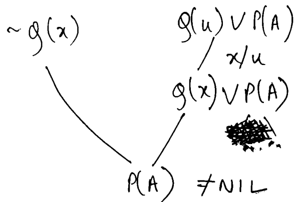
$\sim Q(w) \vee P(w)$

$\sim Q(x) \vee \sim P(x)$

$Q(y) \vee \sim P(y)$

Suppose we try to resolve 3rd & 4th, then with 2nd & then with 1st.





We cannot approach further as ~~there~~ is no single literal clause. Also we cannot backtrack. This approach is non-complete although efficient.

iii) Set of support strategy

At least one parent of each resolvent is selected from among the clauses resulting from negation of goal wff. or from their descendants.
(to reach the goal faster; it ~~is~~ complete)

iv) Unit preference strategy

This is a modification of set of support. Instead of filling out each level in breadth first order, try to select the single literal clause (called a unit ~~&~~ hence the name) to be a parent in resolution.
(Why? → to reduce the no. of literals)

v) <u>Ancestry filtered form strategy</u>

Each resolvent has a parent that is either in the base set or ancestor of other parent.
Contension of (ii)

vi) <u>Combination of strategies</u>

Set of support with either linear i/p form or ancestry filled ~~f~~ form

<u>Answer Extraction Process</u>

<u>Extracting Answers from Resolution Refutation</u>

If Fido goes wherever John goes and if John is at school, where is Fido?

~~Locati~~ <u>Wffs</u>

~~Loc~~

$(\forall x)\{Dest(John, x) \longrightarrow Dest(Fido, x)\}$

$Dest(John, S_1)$

$\sim(\exists y)\{Dest(Fido, y)\} \Rightarrow (\forall y)\{\sim Dest(Fido$

where is Fido?

if we dont give $(\forall x)$ scope of x differs and also, not true for all x.

~~~Dest (John, S_1)~~ ~~Dest(John, x) → Dest~~

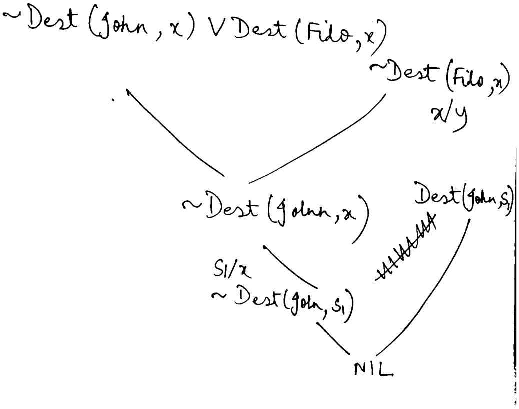
Clause

{~Dest(John, x) V Dest(Fido, x)}

~~{Dest(Fido, A)}~~ {Dest(John, S_1)}

~~{Dest(John, S_1)}~~

{~Dest(Fido, y)}


~~~

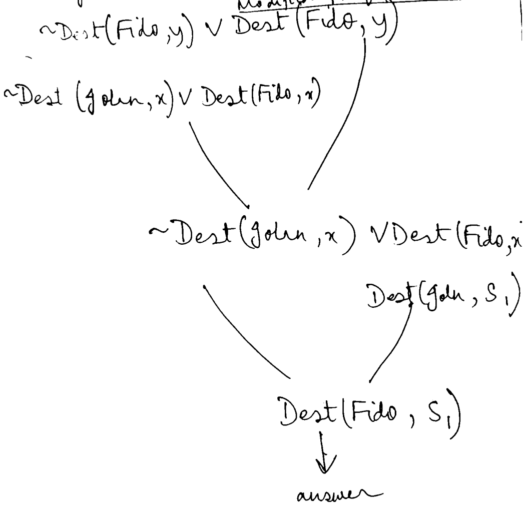
Modified Proof Tree (to find answer):

Add W to ~W

ie W V ~W

Change ~Dest(Fido,y) to

Modified Proof Tree


~Dest(Fido,y) V Dest(Fido,y)
~Dest (John,x) V Dest(Fido,x)
~Dest(John,x) V Dest(Fido,x
Dest(John, S1)
Dest(Fido, S1)
answer

So far we have been doing forward resolution.

We can do backward resolution as well

Soundness & Completeness of a System

↓
from book

Resolution process is sound & RFT is complete.

8/10/15

AND-OR graphs

AND nodes – all of the nodes to be processed
OR " – one of the " " " "

Hypergraphs

arcs – hyperarcs
connectors – ~~hyper connector~~ – K connector

Formulating game playing as search :–

- Two person, perfect info
  players move alternately
  no chance factor
- Search space too large – iterative methods
- Adversary search
- static evaluation fn.

f(n) = large +ve (good for me, bad for opp.)
= large -ve (bad for me, good for opp.)
= near 0 (draw situation)
= + ∝ (winning config for me ∄ very likely)
= – ∝ (losing config for me ∄ very likely)
ev. fn. of Tic-Tac-Toe

Game Trees

- Root → present config

  If my turn → MAX

- ~~Ans~~ Arcs – possible legal moves

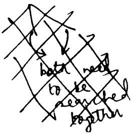
- i → if all MAX
  then i+1 → all MIN & vice versa

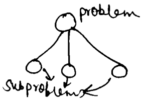
- nodes corresponding to MIN's next move have successors that are like AND no

- nodes corresponding to MAX's next move ~~to~~ have successors that are like OR nodes.

game	# nodes in complete gametree
~~chickens~~	
checkers	$\approx 10^{40}$
chess	$\approx 10^{120}$

OR graph (at any one time, we pick one of n alternati

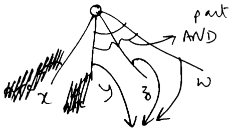




all branches need to be solved independently

AND graph

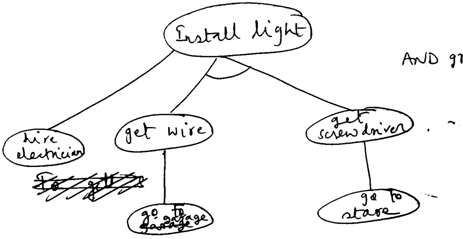
[redacted]



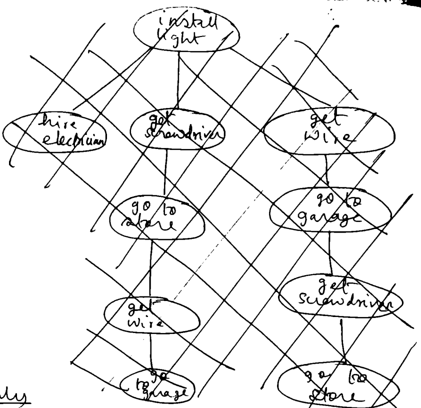
part of graph is AND ie. we need to solve either x or y, z & w together ~~#~~. x || (y && z && w)

K arcs can be combined, K-connector

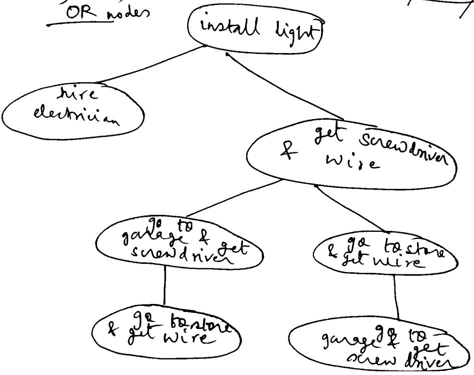
i) If I can get ~~a~~ wire and a screw-driver, I can install the light.



ii) To get screwdriver, go to garage.
iii) To get wire, go to some store.
iv) ~~##~~ If whole thing not possible, hire an electrician.



Using only OR nodes



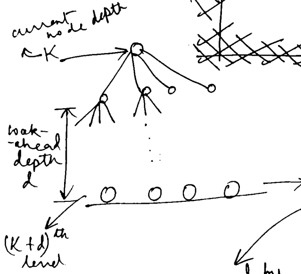
AO* ~~AO*~~ search used in these OR-graphs for searching.

Game playing using adversary search

I make a move that will make me win.

Next step, opponent makes a move that will make me lose (worst config for me).

~~look-ahead depth~~

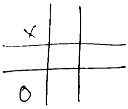


so called 'leaf' nodes, given some 'goodness' value & based on ~~that~~ we will propagate up to my current level & then make a move

measured by evaluation fn. called static eval. fn.

Eval fn. for tic-tac-toe : No. of 3 rows/cols/diag open for me — No. of 3 rows/cols

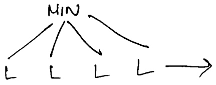
/diag
Open for my opp.



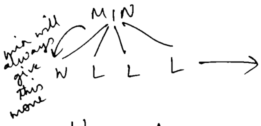
For chess, ~~[illegible]~~ the eval fn. cannot be found so easily bec. each piece needs to be given diff. wt but dont know how mu..

In ttt, both pieces have same weight, so easier to find eval fn..

Game Trees



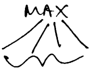
only ~~may~~ then can max win



~~[illegible]~~ so unless I conside all the successor nodes, I will not be able to tell whether I'll win or not. So these nodes will be like ~~OR~~ AND nodes.

W → goodness value in my favor

L → goodness value in opp.'s favor



if there is even one W pos, it goes in my favor, so they are like OR nodes

2 algos

→ MINI-MAX algo

→ α-β pruning $\xrightarrow{algo.}$ extension of mini-max where some branches will be pruned off

MiniMax algo

1) Create start node ~~s~~ as a ~~[illegible]~~ MAX node (my turn to move) with current ~~root~~ board configuration.

2) Expand nodes down to some depth (ply) of lookahead in the game.
↓
to look ahead

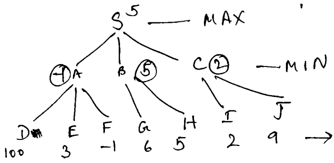
3) Apply evaluation fn. at each leaf node.

4) Back up values for each of the non-leaf nodes until a value is computed for the root node.

At MIN nodes, backed up value is the minimum of the values associated with its children.

At MAX nodes, " is max of
" " "

5) Pick the operator associated with the child node whose backed up value detemined the value at the root.



→ these values are predicted values as we cannot ~~to~~ see beyond a certain depth

estimated goodness val.

S → B → H

Drawback

This algo is exponential in terms of time complexity

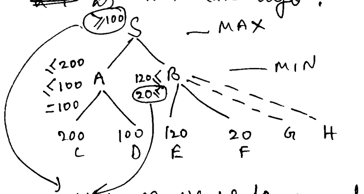
$b^0 + b^1 + b^2 + \ldots$

α-β pruning

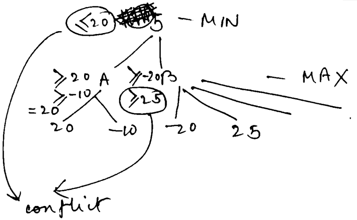
Pruning ~~done~~ done to prune off branches which are "blind lanes".

→ In chess, 35 branches reduced to 6 branches.

→ α-β pruning gives the same goodness value ~~α-β~~ as minimax algo.



conflict, so we no longer need to search ~~for~~ ~~the~~ through the remaining branches.



Parent is MAX— α pruning
MIN— β "

The stored value at non-leaf node is either α or β.
AT MAX, we call it α
AT MIN, " " it β.
∴ α= best value found so far at a MAX node
β = best value found se far at a MIN node.

~~α=~~
α-value of a MAX node is monotonically non-decreasing.
β- value of a MIN node is monotonically non-increasing

β~~α~~- cut off

Given a node n, cut off the search below n, ie do not generate any more of n's children if:
n is a MAX node & α(n) ≥ β(i) for some MIN node ancestor i of n

α- cut off

.... if :

n is a MIN node & $\beta(n) \leq \alpha(i)$ for some MAX node ancestor i of n

Reduces no. of nodes examined
Gives same goodness value as min-max

$2b^{d/2}$ → In best case, no. of leaf nodes examined
[d → look ahead depth, b ≡ branching facto

b^d → in worst case

Best case occurs when:
MAX node → child with the largest value is examined first
MIN node → child with the smallest value is examined first

Game Playing ~~Algorithm~~ program

Contains
- Smart opponent — no oversight error, no psychological factor
- Dumb terminal

2 types of game playing programs—
Type A → It's a dumb fast evaluator, lots of look ahead search
Type B → It's a smart slow evaluator;

α- cut off

.... if :

n is a MIN node & $\beta(n) \leq \alpha(i)$ for some MAX node ancestor i of n

Reduces no. of nodes examined
Gives same goodness value as min-max

$2b^{d/2} \rightarrow$ In best case, no. of leaf nodes examined
[d → lookahead depth, b ≡ branching factor

$b^{d} \rightarrow$ in worst case

Best case occurs when:
MAX node → child with the largest value is examined first
MIN node → child with the smallest value is examined first

Game Playing ~~Algorithm~~ program

Contains
- Smart opponent — no oversight error, no ~~pt~~ psychological factor
- Dumb terminal

2 types of game playing programs—
Type A → It's a dumb ~~[illegible]~~ fast evaluator, lots of look ahead search
Type B → It's a smart slow evaluator;

little search

Type A → machine
" B → human

Horizon Effect

At depth d+1, situation may ~~be~~ reversed, we may get an ~~even~~ better soln. But it is beyond ~~o~~ our horizon.

14/10/15

Uncertainty Handling

info is not complete/uncertain:
answers with uncertainty → how we represent them & handle them.

Predicate Logic

In PL, we are given certain/complete/definite facts, info, rules & we can draw confident conclusions

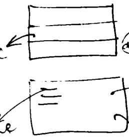
Non-Monotonic Reasoning System

for uncertainty handling

Refer Artificial Intelligence by Patterson

Nilsson (earlier ed.) → Prediate logic
Norvig → Search algos.

ansmitaghoshju@gmail.com → send ma'am scans & remind ma'am about class on 29th



monotonic

absolute truth.

we believe these as we don't have info to the contrary

beliefs

does not rem. true forever

if some info here gets deleted, some info dependent on it may need to be deleted. So siz of 'beliefs' database decreases & hence real-life situations are non-monotonic in nature

revising beliefs → addition of beliefs/
deletion ,, ,,

database should be consistent at any time t so that we can draw inferences

belief	justification
——	- -
——	- -

dependency record

for each belief, there is its justification

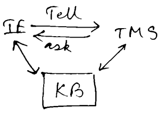
Methods to deal with uncertainty—

i) TMS— truth maintainence system
ii) Modal & temporal logic
iii) Fuzzy logic
iv) Reasoning based on prob.

TMS
↓
allows addition of changing (even contradicting) ~~statements~~ beliefs to knowledge base.

Belief revision → maintains consistency

TMS → simply maintains consistency.
IE → inference engine, draws inferance



TMS maintains dependency record, it uses support list justification (SL)

contains for:
Innodes → all in-nodes for which it is true
Outnodes → all outnodes for which it is true

SL (in nodes) (out nodes)

1) It is sunny. SL (2) (4)
have to be true → 2; must not be true → 4
2) It is day time. SL (NULL) (NULL) → abs. truth.
3) It is raining. SL (NULL) (1) → may be 2
4) It is warm. SL (1) (null)

~~SL(1) → NULL~~
~~SL(2) → 1~~
~~SL(3) → NULL~~
~~SL(4) → 1, 2~~

~~SL(1)~~

Premice : in-lict , out-lict both empty
Normal derivation : outlist always empty.
Ascumption : out-lict never empty

CP → conditional proof justification

CP <consequent> (in-hypothesis)
~~CP → out~~
CP also includes out-hypothesis — always empty

consequent is IN
if all nodes in in-hypothesis are IN