Overview of Soft Computing Methods in Multidisciplinary Design

P. Hajela
Rensselaer Polytechnic Institute
hajela@rpi.edu

Soft Computing in Design

- Simulated annealing (SA) and genetic algorithms (GA's) - search in mixed design space without gradient information
- Neural networks - function approximation, identifying causality from numerical data, control system synthesis, combinatorial optimization tools
- Immune network modeling - decomposition based design, enhancing efficiency of GA based search, multicriterion design
- Fuzzy logic - modeling of manufacturing processes, design with non-crisp information
- Intelligent-Agents - human interface issues
- Machine learning paradigms - function modeling, deriving context specific rules

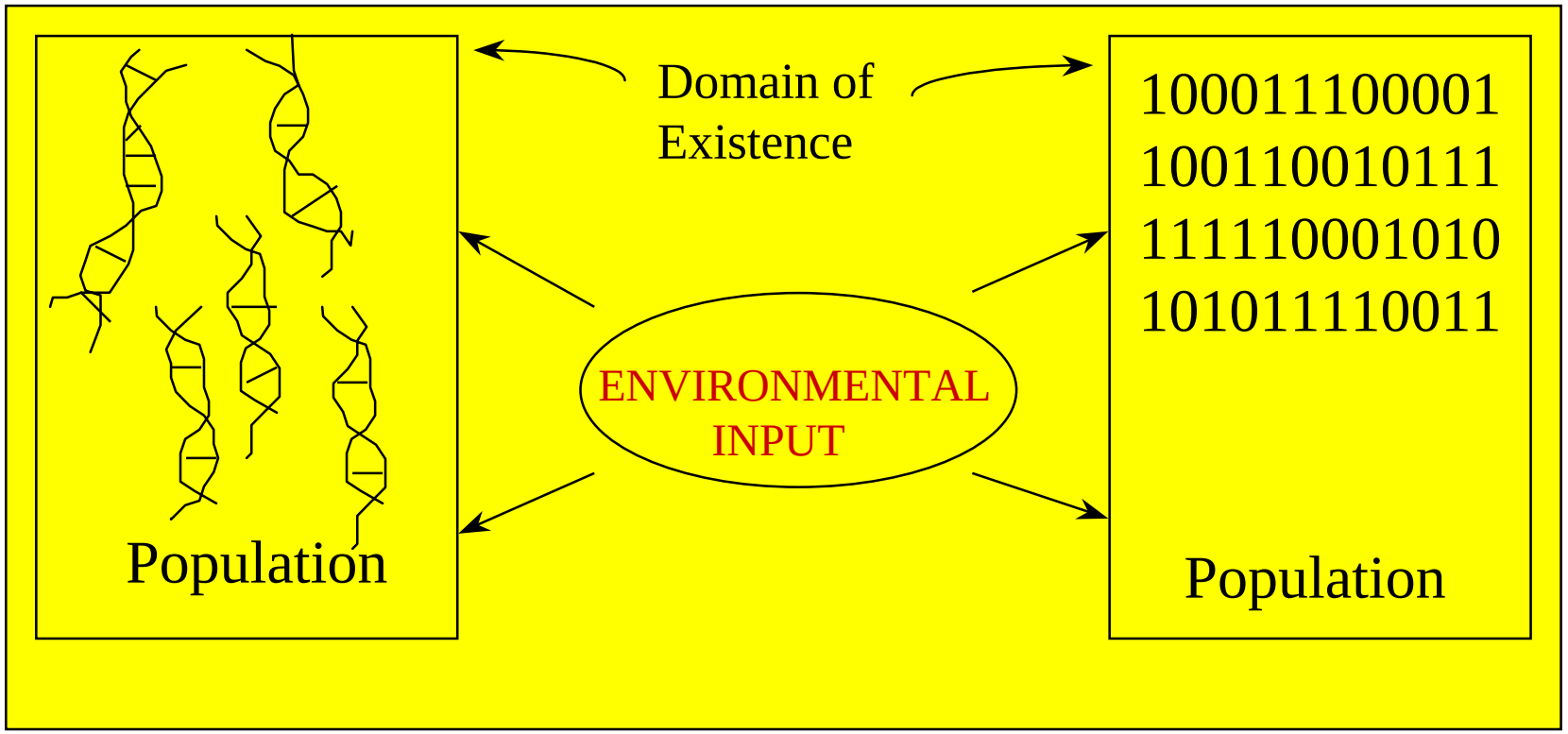
Genetic Algorithms

- Philosophical basis of search methods is in Darwin's theory of "survival of the fittest”
    - belongs to a general category of stochastic search methods - random choice is used as a tool to guide a highly exploitative search
- Early ideas due to Rechenberg (60's); contemporary format as proposed by John Holland (1975); contributions of Davis, DeJong, Goldberg and Grefenstette useful in obtaining a better understanding of approach
- Now extensively studied and used in engineering design

GA Based Optimization

- Global search strategies like GA's offer improved performance over traditional mathematical programming
- Populations of designs, each represented by a chromosome-like string structure are manipulated in a manner analogous to biological evolution
  - global search using information from different regions of the design space - global vs. relative optimum
  - easily implemented for mixed variable problems
  - requires no gradients

Evolutionary Analogy



Coding in Genetic Search

- Map a variable X into a chromosome-like string. Assume that X can take on values from the following discrete set.

  X = 1 3 4 6 7 8 11 14

- A 3-digit binary string offers eight possible combinations that can provide such a mapping capability

000 - 1	001 - 3
010 - 4	011 - 6
100 - 7	101 - 8
110 - 11	111 - 14

Representation of Discrete/Integer Variables

- Discrete nature of binary representation - ideal for discrete variables; integer variables are discrete variables with spacing of unity
- The required string length p is computed as follows

$$2^p \geq \frac{\left(X^U - X^L\right)}{A_c} + 1, \text{ where } A_c \text{ is precision}$$

  - for discrete variable with k alternatives, $2^p \geq k$

- Equality - one-to-one correspondence
- Inequality - nonunique representation of design variables

Inequality Handling

- Penalty Approach
  - Compute smallest p that satisfies the inequality

$$2^p \geq \frac{\left(X^U - X^L\right)}{A_c} + 1$$

    - assign unique string to each m discrete variables
    - assign remaining strings to out-of-bound integers
- Excessive Distribution Approach
  - Assign excessive bit-strings to already assigned variables
    - uneven expansion of the design space - could target a string length that would give a more even expansion of the design space

Non-binary Representations

- Note that binary coding is not essential to the implementation of genetic algorithms - use of real variables has been explored
- Consider the design of a composite laminate for its stacking sequence - admissible angles are 0 deg, 45 deg, -45 deg and 90 deg.
- Represent these angles by 1, 2, 3 and 4, respectively
  - a 16-ply laminate $(+45/-45/0_4/\ 90_2)_S$ is represented by a code (2311114444111132)
  - symmetry can be assumed and an 8-digit string used instead (23111144) - reduction in the number of possible designs from 4^{16} to 4^8

Design Variable Mapping

- Map design variables into finite-length strings - this mapping must proceed under some guidelines
  - use smallest alphabet for coding the problem - larger alphabet suppresses similarities that are exploited by the GA search
- Consider a variable X that can take on integer values between 1 and 26 - map using a 5-digit binary string and the alphabet A-Z.

X	f=X^2	Binary Representation	Non-Binary
8	64	01000	H
12	144	01100	L
24	576	11000	X
19	361	10011	S

Design Variable Mapping

- If a string length p is used to represent a design variable x, we have available integers from 0 to 2^p - hence to represent design variable values in this range, use a linear map of
  - » $[0, 2^p] \rightarrow [X_{min}, X_{max}]$
- Assume p=5, then X_{min}=00000 and X_{max}=11111
  - the precision of mapped coding is computed as Ac
    - » $Ac=(X_{max}- X_{min})/(2^p -1)=0.0323(X_{max}- X_{min})$
- Intermediate values of binary numbers obtained as a simple linear scaling process
- Note that precision of mapping may be enhanced by reducing the range of the mapping ($X_{max}- X_{min}$), or by increasing string length p

Mechanics of Genetic Search

To mimic the biological evolution process, there is a need to define

- A population of designs that must evolve from one generation to another.
- An environment to which the population must adapt - all measures of fitness of design, including objective function and constraint values.
- The basic operators which transform a given population into one that is better adapted to the environment - higher average value of fitness.
  - Reproduction
  - Crossover
  - Mutation

Mechanics of Genetic Search

Reproduction

- Biases the evolutionary process towards more fit members of the population - is implemented by adding extra copies of more fit members into the population pool.
  - no new designs are created during this operation.

Crossover

- Is the mechanism by which new information is brought into the population pool - akin to exchange of genetic information between mating parents.

Mutations

- Is a random switching of bits in a string at randomly selected sites (very few!) to prevent premature loss of genetic information. This becomes necessary due to the finite-size populations used in practical simulations.

Fitness Definition

- Consider a function maximization problem

$$Maximize\ f(X)$$
$$X^L \leq X \leq X^U$$

- For function minimization, following choices of a fitness measure can be adopted.

  F = 1/f(x)

  $F = F_{max} - f(x)$

- For the treatment of design constraints, the fitness function must be appropriately modified
  - **penalty function formulation is traditionally employed - sensitive to the relative magnitude of the objective and the constraint term. Alternative "expression operator" based approach proposed.**

Basic Operations

A: Reproduction

- Assign each design i, a probability of selection as $p_i = \frac{F_i}{\sum_N F_i}$

- At random, select the N most fit designs for the crossover and mutation operation.

B: Crossover

- Of the N-member population generated during reproduction, select mating pairs, two at a time, to conduct the crossover operation

Basic Operations

Parent 1: 1100010011
Parent 2: 1011011011
Child 1: 1100011011
Child 2: 1011010011

- Repeat for all possible mating pairs
- Note that real variables can also be used to code design variables into stringlike representations

Basic Operations

- Consider the design of a rectangular laminated plate, where plate dimensions are to be sized in addition to determining number of 0_2^o, +/-45^o and 90_2^o groups

  parent 1: 10.1 7.3 5 0 0 is a 10.1x7.3 plate of layup $(0_{10}^o)_S$

  parent 2: 12.5 6.4 2 1 2 is a 12.5x6.4 plate of layup $(0_4$,+/-45, $90_4^o)_S$

- In such representations, child design will only contain numbers that were present in the original population - one solution lies in an averaging crossover

  – for a string length L, generate a real random number w=rL between 0 and L - define [w] as integer portion of w

Basic Operations

- take first [w] variables from one parent, last L-[w]-1 from second parent, and average the [w]+1 variable between two parents
- if x^1 and x^2 are the [w]+1 variable of parent 1 and 2, average this number to create x^c for child as

$$x^c = (w - [w])x^1 + (1 - w + [w])x^2$$

- For w=1.6, [w]=1, and child is generated by using first number from parent 1, last 3 from parent 2 and averaging the second number between two parents as

$$x^c = (1.6 - 1)7.3 + (1 - 1.6 + 1)6.4 = 6.9$$

$$\text{child}\,1 = 10.1 \;\; 6.9 \;\; 2 \;\; 1 \;\; 2$$

Basic Operations

C: Mutation

- Select strings at random from the population pool, and at randomly selected sites, switch the zero to one or vice versa.
- The crossover and mutation operators are assigned a finite probability $p_c \sim 0.7$ $p_m \sim 0.01$

Basic Operations

D. Permutation

- An operator referred to a permutation has been explored with some success in some recent studies. Permutation may be viewed as a mutation that is applied over a segment of the string.
- Balanced $[45/-45/0/90_2/0/45/0_2/-45]_S$ balanced, symmetric laminate

  before permutation 231<u>4412</u>113

  after permutation 231<u>2144</u>113

  resulting in a $[45/-45/0/45/0/90_2/0_2/-45]_S$ laminate

GA’s vs Traditional Methods

- GA's work with a coding of the entire set of design variables and not the variables themselves.
- GA's do not optimize the design by advancing it from point to point. Instead, they work from a population of designs, advancing several designs in each cycle of evolution.
- Only function information is required - no gradients.
- Evolution and adaptation are implemented by nondeterministic transition rules.
- Implicit parallelism embedded in approach which yields significant computational advantage.
  - n evaluations of a population really results in the exploration of n^3 schema

Constraint Handling in GA's

- Genetic transformation operations require that a fitness measure be computed for each design. This measure is maximized during the search process.
  - for unconstrained maximization, the objective function itself may be chosen as a fitness measure.

$$Z = F(x)$$

  - for unconstrained minimization, the fitness measure may be defined as

$$Z = F_{max} - F(x)$$

- Constrained minimization, composite measure of objective and constraint functions be used in defining fitness function

Constraint Handling
Penalty Formulation

$$Minimize\ \overline{F} = F(X) + \overline{P} \quad and \quad Z = \overline{F}_{\max} - \overline{F}$$

- $\overline{P}$ is the penalty term - must be chosen carefully so as to prevent biasing the search in favor of objective or constraint functions.
- For an average fitness of feasible designs as F_{ave}, define a limit value of the penalty $\overline{L} = kF_{ave}$ where $k \cong 2$

$$\overline{P} = \begin{cases} G & \text{if } (G \le \overline{L}) \\ \overline{L} + \alpha\,(G - \overline{L}) & \text{if } (G > \overline{L}) \end{cases} \qquad G = r\sum_{j=1}^{m} \langle g_j \rangle$$

- where r is the penalty parameter, and $\langle g_j \rangle$ represents the violated constraints.

Constraint Handling - Penalty Formulation

- Scaling prevents constraint violations from biasing the search process.
  - If α=0, penalty of all violated constraints is limited to $\overline{L}$
  - For α=0.2 or similarly small value, constraint violations are allowed to increase beyond $\overline{L}$, albeit with a small slope.
- Performance of the penalty function approach is clearly dependent upon user specified constants such as penalty parameter r, factor k used to establish $\overline{L}$, and slope parameter α

Expression Based Strategies

- In the biological analogue of genetic search, the chromosome is a double-stranded (diploid) structure - expressed gene at a particular location is determined on the basis of a dominant-recessive gene theory.
  - in a traditional GA implementation, the designs are represented by a single stranded (haploid) structure.
- Combine features of feasible and infeasible designs in a given population through the use of an expression operator - probabilistic in nature.

Expression Based Strategies

- Temporarily assembled diploid model
  - String A (feasible) : 1110010110
  - String B (infeasible): 1000011011
- “Expression” operator applied on a bit-by-bit basis to determine an expressed chromosome that would replace string B in the population.

  replace bit at a location in string B by corresponding bit on string A with probability p_E.

Expression Based Strategies
Stepwise Implementation

- (Step 1) Population is initialized - uniform normal distribution of design variables between specified lower and upper bounds.
- (Step 2) Evaluate population of designs to evaluate objective and constraint function values.
- (Step 3) Combine infeasible and feasible designs in the population through use of the expression operator - two strategies explored.
- (Step 4) Determine objective and constraint function values of expressed designs.
- (Step 5) Selection operation of traditional GA applied to population of feasible designs. Selected designs combined with all infeasible designs to obtain predetermined population size.
- (Step 6) Perform crossover and mutation as in a traditional GA implementation.
- repeat from step 2 to convergence

Expression Based Strategies - Strategy I

- Identify best feasible design in population as X_{best}.
- Rank all j infeasible designs in population (j=1,N) as R_j, with R_j=N assigned to design with most constraint violation.
- Combine each infeasible design with X_{best} through the use of the expression operation on a bit-by-bit basis

$$g_i^E = \begin{cases} g_i^B & \text{if } (r_i^N < R_j) \\ g_{ij}^V & \text{if } (r_i^N \geq R_j) \end{cases}$$

Expression Based Strategies - Strategy I

g_i^E : i-th bit on chromosome representing expressed design.

g_{ij}^V : i-th bit on chromosome representing j-th violated design.

g_i^B : i-th bit on chromosome representing best design.

r_i^N : randomly generated integer between 1 and M (population size).

- Designs with higher constraint violations become similar to chromosome of best design.

Expression Based Strategy II

- Selection of feasible-infeasible pair determined by closeness in terms of the objective function value.
- If δ_{IJ} = obj(I) - obj(J) is difference in objective function values between an infeasible design J and a feasible design I, design I with smallest absolute δ_{IJ} is chosen for expression.
  - Negative δ_{IJ} preferred over positive value for comparable absolute value of δ_{IJ}.

Expression Based Strategies - Strategy II

g_i^E : **i-th bit on chromosome representing expressed design.**

g_i^{BF} : **i-th bit on chromosome of feasible design selected for expression.**

g_{ij}^V : **i-th bit on chromosome representing j-th violated design.**

r_i : **random number (uniform distribution) between 0 and 1.**

p_E : **fixed probability of expression**

$$g_i^E = \begin{cases} g_i^{BF} & \text{if } (r_i \le p_E) \\ g_{ij}^V & \text{if } (r_i > p_E) \end{cases}$$

Simultaneous Min/Max Identification Sharing Functions in GA

- Based on a principle of sharing available resources of an environment to maximize individual gains for distinct species
- Sharing implemented by degrading fitness of each design in proportion to the number of designs located in its neighborhood

$$\phi_{ij} = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{sh}} \right)^{\alpha} \\ 0 \end{cases}$$

- d_{ij} is the metric distance between the *i*th and *j*th design, and σ_s is the radius of a defined neighborhood

Sharing Functions in GA

- Fitness of each design is modified as follows

$$f_{si} = \frac{f_s}{\sum_M \phi_{ij}}$$

  - M is the number of designs in the neighborhood of design I
- Sharing function approach has been incorporated in multicriterion optimal design

Simulated Annealing

- Qualitatively derived from the behavior of particles in thermal equilibrium at a given temperature — Metropolis and co-workers.
- Kirkpatrick was the first to use such a simulation in application to a combinational optimization problem
- As in genetic search, method belongs to the general category of stochastic search techniques.
  - requires a higher investment of computational resource then traditional mathematical programming
  - offers a greater potential for locating global optimum

Simulated Annealing

- Has its basis in principles of statistical mechanics
  - consider the atoms of a molten metal at some highly elevated equilibrium temperature — state of system ϕ is characterized by specific spatial locations of atoms
- Probability $P_T(\phi)$, that at a given equilibrium temperature, the system is in a state ϕ, is given by the Boltzmann distribution

$$P_T(\phi) = \frac{\exp(\frac{-E(\phi)}{KT})}{\sum_{\Phi} \exp(\frac{-E(\phi)}{KT})}$$

K: Boltzmann constant

E(ϕ): Energy of state ϕ

T: Temperature

Φ: All possible states at a given temperature

Simulated Annealing

- At a given temperature, random variations of the system state are considered.
  - If a state results in a lower energy level, it is immediately accepted
  - If a higher energy state results, it is only accepted with a probability

$$p = \frac{P_T(\phi_2)}{P_T(\phi_1)} = \exp[(E(\phi_2) - E(\phi_1)]/KT, \quad E(\phi_2) > E(\phi_1)$$

  - Note that with decreasing temperatures, probability as defined above gets lower, indicating that higher-energy level states are more likely to be rejected at lower temperatures.

Stepwise Approach

- State of system defined by values of the system design variables - assume that this state is denoted as x_1.
- Compute the analogous "energy" $E(x_1)$ corresponding to "state x_1" - this can be the objective function value with the constraints appended as a penalty.
- At some initial high temperature, state x_1 is varied at random to obtain new state x_2 for which $E(x_2)$ is computed
  - if $E(x_2) < E(x_1)$, then x_2 is accepted immediately as a new state of system
  - if $E(x_2) > E(x_1)$, then x_2 is accepted only by probability obtained as ratio $p_T(x_2)/p_T(x_1)$

Simulated Annealing

- Process of design variation done enough times to ensure equilibrium at a given temperature
  - in practical terms, this translates into allowing a fixed number of random variations at a given temperature
- Temperature is lowered by an amount specified in the "annealing schedule" and random variations are repeated.
- Process carried through to convergence for a final prespecified value of the annealing temperature

Simulated Annealing

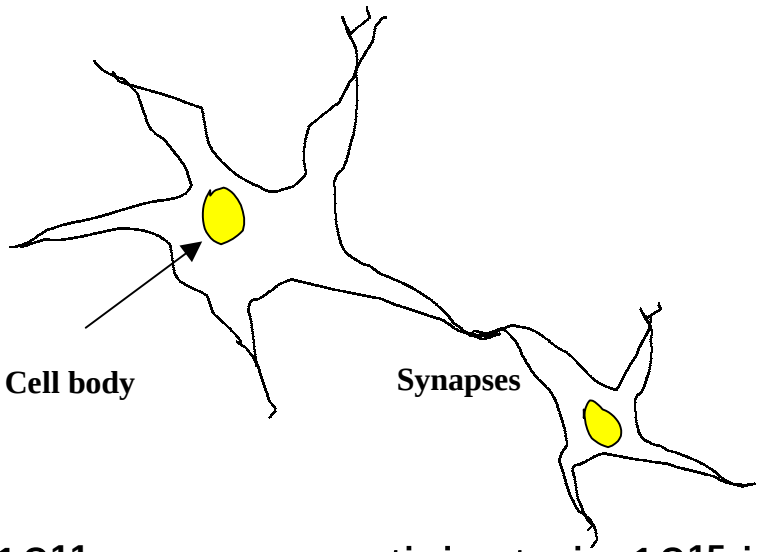
- Traditional optimization methods may be considered as a rapid rate quenching -- at best an ordered state with some low, but not necessarily lowest energy state will result.
  - Slower temperature reduction schedule and higher number of function evaluations imply higher computational costs.
  - Easy to accommodate discrete and integer design variables.
  - Only function evaluations are required -- no gradient calculations.
- Ability to accept poorer designs, albeit with lower probability provides a mechanism to escape from locally optimal design points.

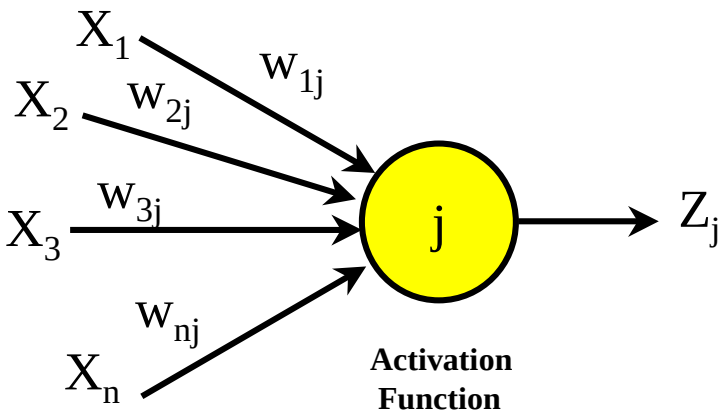
Neural Networks

- Applications of neural networks in multidisciplinary design
  - function approximation ideas
  - for determining decomposition models and accounting for interactions among temporarily decoupled subsystems
  - as optimization tools
- Of these, the function approximation idea is the most widely explored; two models of neural networks will be presented
  - backpropagation network (BPN)
  - counterpropagation network (CPN)

Neural Networks

- Artificial neural networks are biologically inspired. Despite expectations of brainlike performance, the resemblance of such networks to their natural counterpart is at best, superficial.





- 10^{11} neurons participate in 10^{15} interconnections in an average biological system

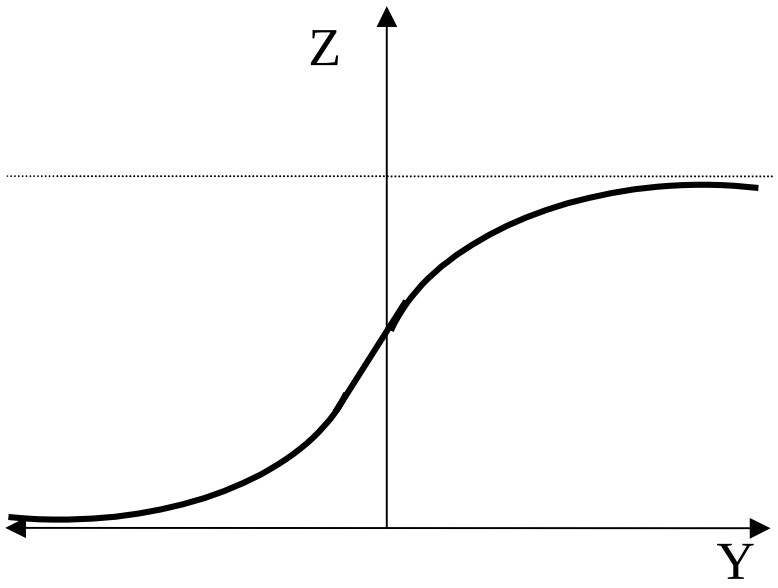
Single Neuron - Processing Unit

- All inputs to a neuron j are summed as

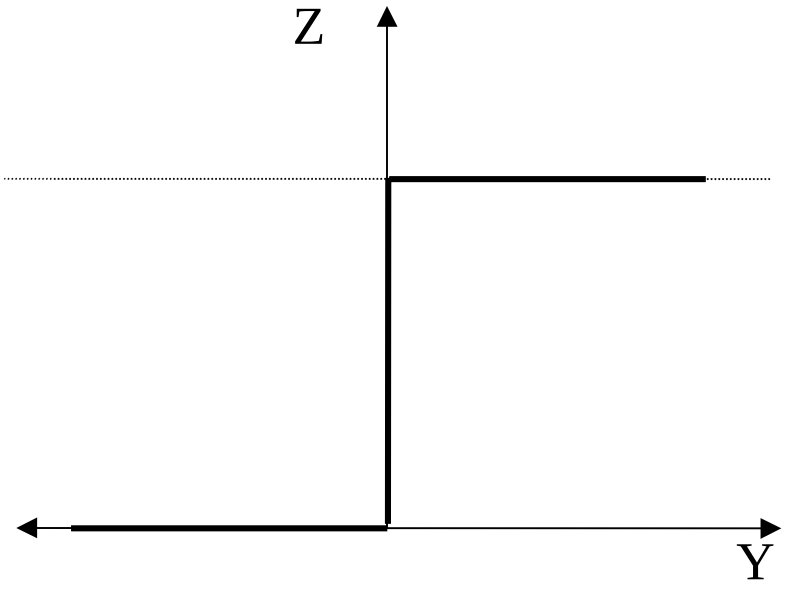
$$Y_j = \sum_{i=1}^{n} w_{ij} x_i$$

- Output $Z_j = F(Y_j)$, where F is referred to as an activation function
- Commonly used activation functions
  - threshold
  - sigmoid
  - hyperbolic tangent

Activation Functions

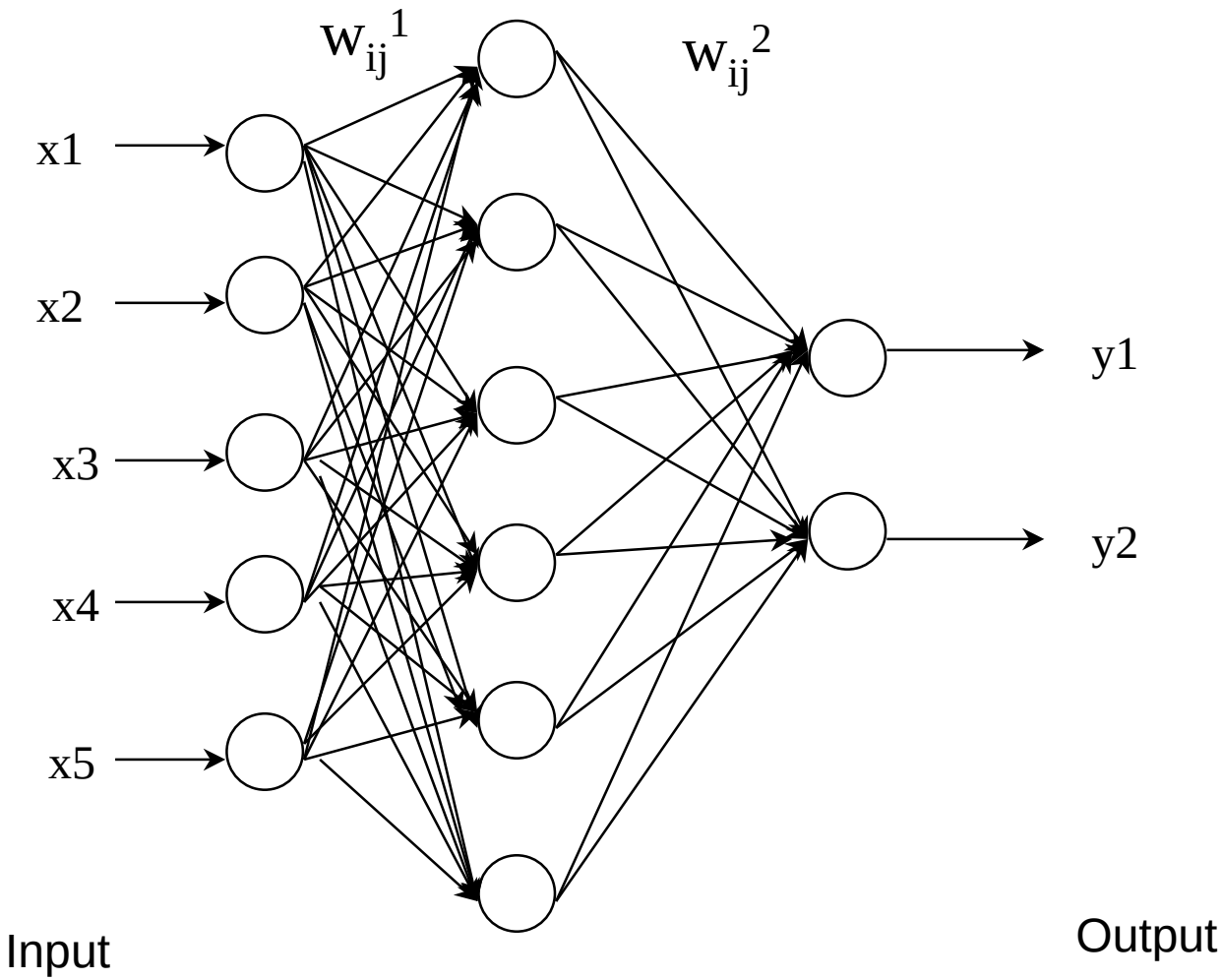


Sigmoid



Threshold

Back-propagation Neural Network



Multilayer Perceptron

- A set of connected layers of neurons comprises the multilayer perceptron architecture
  - one or more hidden layers facilitate nonlinear mappings
- Given a set of input-output data (x-y pairs), obtain values of interconnection weights w_{ij} and activation function parameters to map this data
  - is a specialized form of response surface where order of interpolation polynomial is not required to be stipulated
- Given some x-vector that is not part of the training set, the network should produce an approximation to the output vector y

BP Algorithm - Training

- Initialize weights w_{ij}^{k} randomly
- Compute weighted sum of inputs to a neuron j, and process Y through the neuron activation function

  $$Z = F(Y) = 1/[1 + \exp(-\{Y+T\}/To)]$$

- Compute error in output

  $$E_i = T_i - Z_i$$

- Modify the weights to minimize the error function

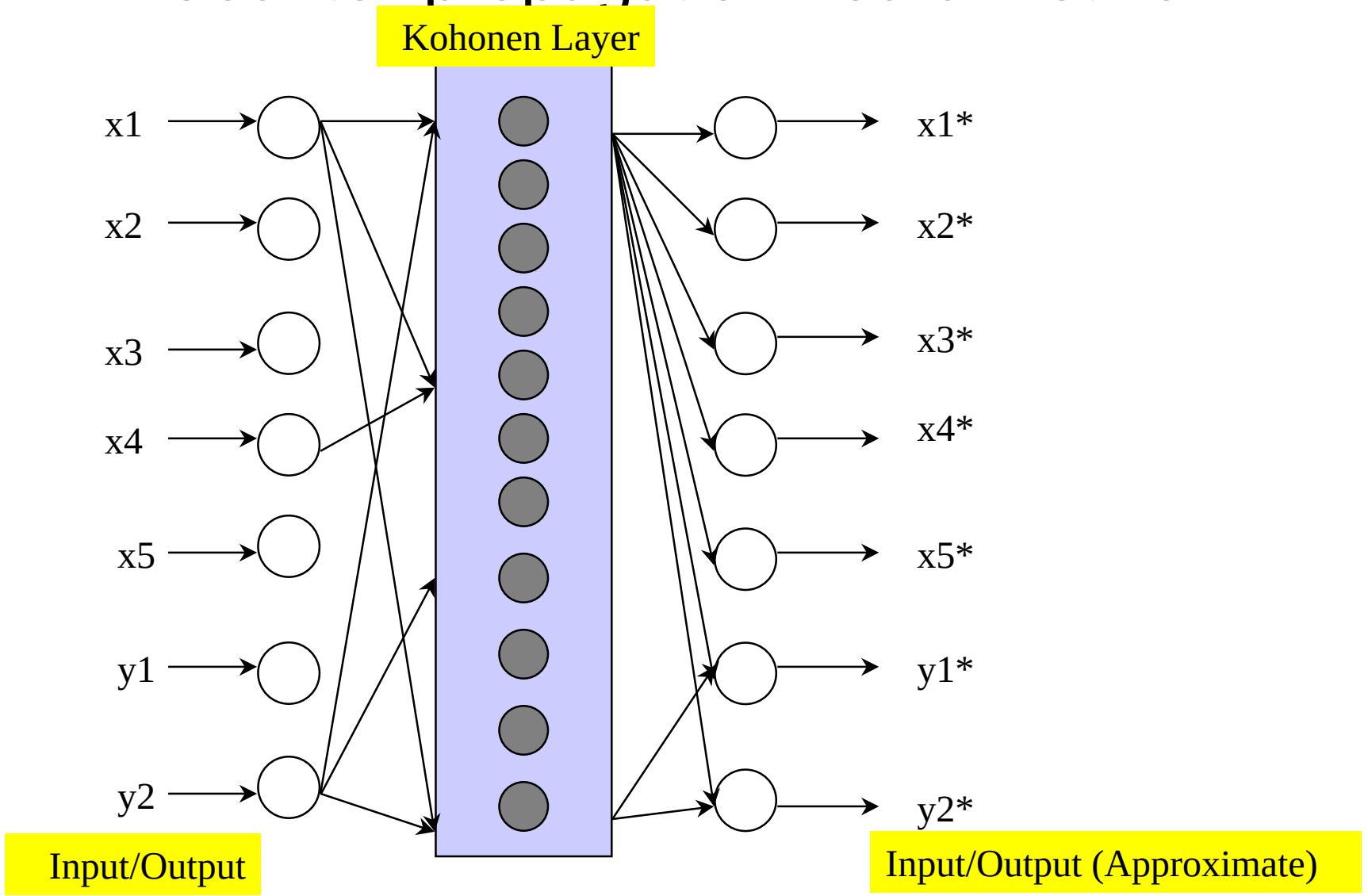
Comments

- Multilayer networks can approximate arbitrarily well given continuous functions, provided that an arbitrarily large number of units is available - property shared by algebraic and trigonometric functions (Weierstrass property)
- Feedforward networks are equivalent to a parametric approximating function f(W,x), where W represent the set of weights in the network.
- Weierstrass property is not very useful for characterizing approximation schemes - the “best approximation” property is what should be required of a network.
  - Regularization networks

Training Tips

- Minimize the number of hidden layers in the network - more hidden layers simply increase the number of weights to be trained.
- Do not “over-train” the network - overtraining would be similar to fitting a high order polynomial through few points
- When training, if error appears to be decreasing at slow rate, randomly “shake” weights to improving training pace
- When training vectors (input-output) are of high dimensionality, see if smaller sub-networks might not be as good as one large network.

Counter-propagation neural network



Counterpropagation Neural Network

- Goes beyond the representational limits of single-layer networks - when compared with backpropagation, significant reductions in training times are obtained.
- Are a combination of self-organizing Kohonen neurons and Grossberg classifiers.
- Network functions as a look-up table capable of generalization - such networks may better represent the brain’s architecture.
- Generalization capability allows the network to produce a correct output even when it is given an input vector that is partially incomplete - pattern matching, pattern completion.

Counterpropagation Neural Network

- Network trains much more rapidly than the BP network
- Network can be used with success in pattern completion tasks
- Network also provides an inverse mapping capability
- Accuracy of generalization has been improved by classifying a given input as belonging to more than one Kohonen neuron, albeit to varying levels.
- A nonlinear averaging procedure is then used to output the response of the network.
- Network size is dynamically adjusted according to the level of precision that is required from the network.

MDO Technology Enhancements Through Soft Computing Applications

Prabhat Hajela

Mechanical Engineering, Aeronautical Engineering & Mechanics

Rensselaer Polytechnic Institute, Troy, New York, USA

Applications

- Use of GA's in generically difficult design optimization problems
  - increased cost in high-dimensionality problems
  - enhanced efficiency of search
  - decomposition based design
- Use of artificial neural networks as function approximators
  - determine causality in data
  - decomposition based design - solution coordination
- Machine learning and computational intelligence

GA Based Optimization - Drawbacks

- Increase in required number of function evaluations to design
  - nonlinear coupled analysis
  - analysis is outcome of physical experiments
- Long string lengths require an increase in the number of design alternatives to be explored
  - increase in design variables
  - increase in number of admissible values of design variables

Enhancements in GA Based Search

- Adaptive strategies for associating significance with bit locations on string
  - directed crossover
  - multistage search
  - immune network modeling
- Decomposition based design
  - partition problem into a number of loosely connected subproblems
  - solve subproblems in parallel (reduced dimensionality)
  - coordinate the solution among subproblems

Biological Immune System

- Simulates the schema recognition/adaptation process central to generation of antibodies in the immune system
- Can be modeled effectively in genetic algorithms with bit-strings replacing chromosomal structure
- Antibodies generated when exposed to the presence of antigens - antigen recognition task must be cooperatively partitioned among gene segments available in the genome
- GA can be used to develop this characteristic

Biological Immune System

Antigen: 11001001011010011010

Antibody: 00110101010010100101

111111xxxx1xxx1111111

XOR Match Score = 13

- Use match score as a fitness function in a GA simulation to develop antibodies to cover specific antigens
- Possible to simulate development of
  - specialist antibodies
  - generalist antibodies

Generalist vs. Specialist

- Immune network simulation to generate both specialist or generalist antibodies
  - consider a simulation with antigens as follows

    000 011 110
  - a generalist antibody would be - 010
    - this antibody is different from each of the antigens by exactly one bit
  - specialist antibody evolution would produce multiple copies of each of the above antigens

Applications of IN Simulation

- Convergence enhancements in GA search - generalists
- Multicriterion design - specialists
- Co-ordination in decomposition based design - generalists

Multicriterion Design

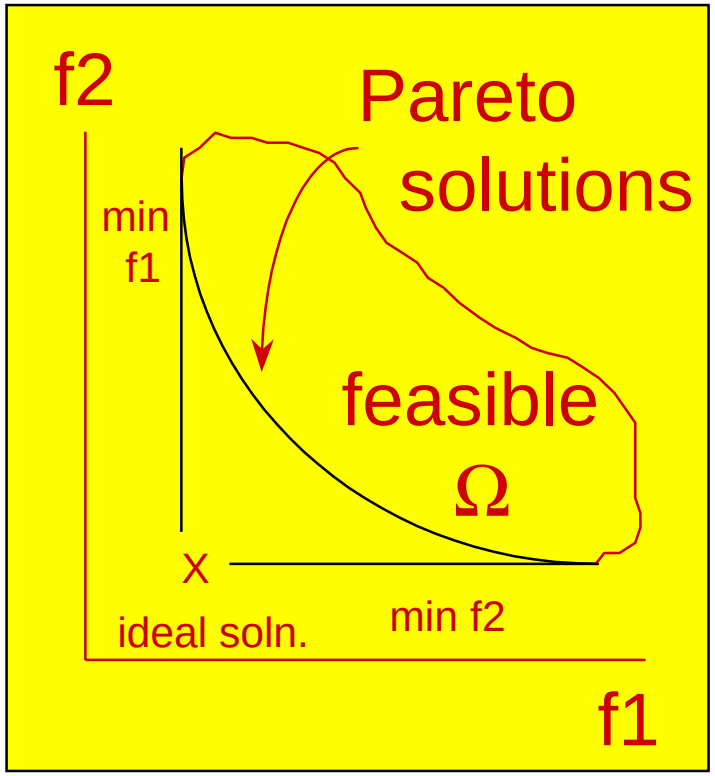
$$\min_{X \in \Omega} f(X)$$

$$f(X) = \{f_1(X), f_2(X), \ldots f_m(X)\}$$

$$\Omega = \{X \in R^n \mid g(X) \le 0, h(X) = 0\}$$

$$f(X) = w_1 f_1 + w_2 f_2 + \ldots + w_m f_m$$

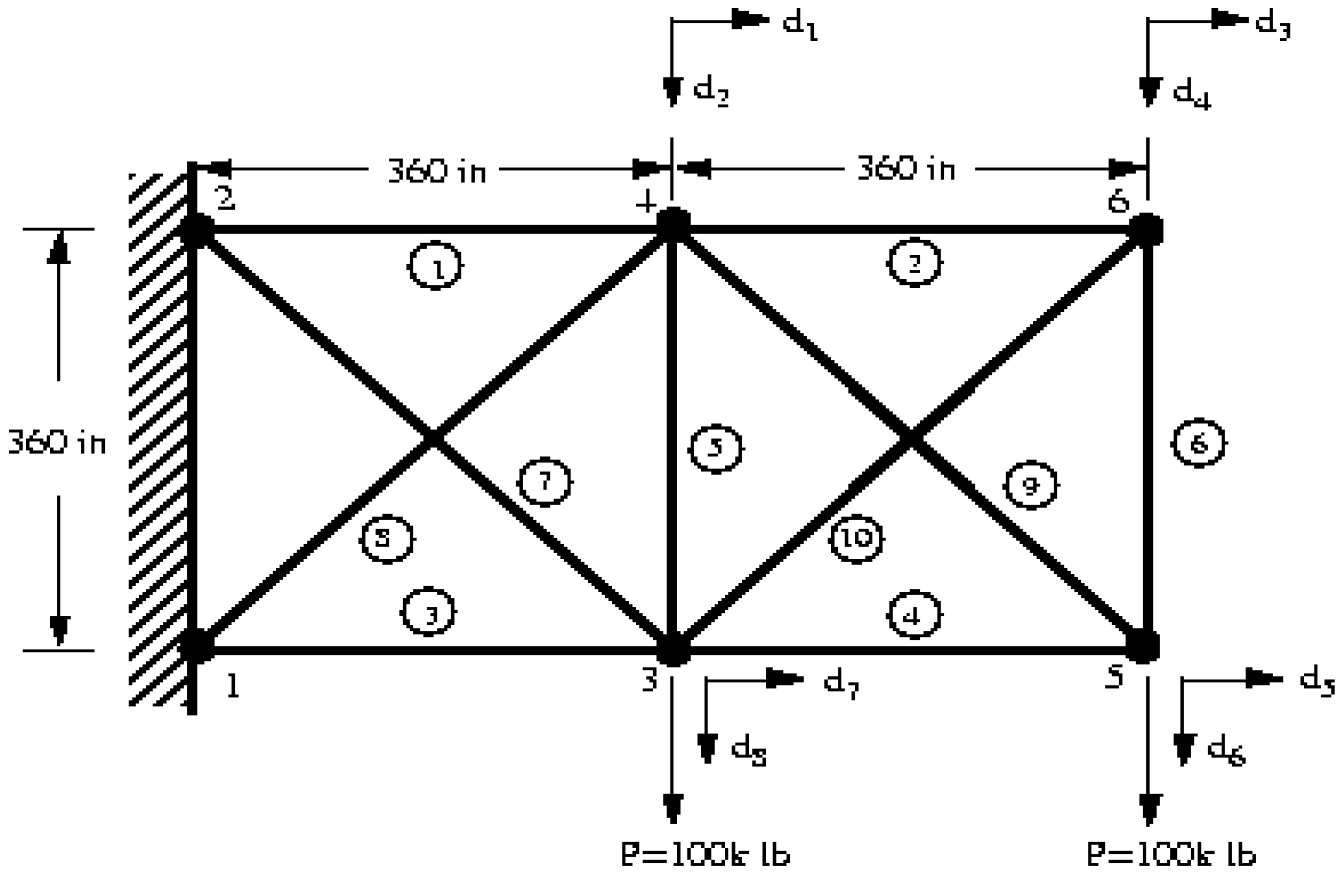
$$\sum w_i = 1$$



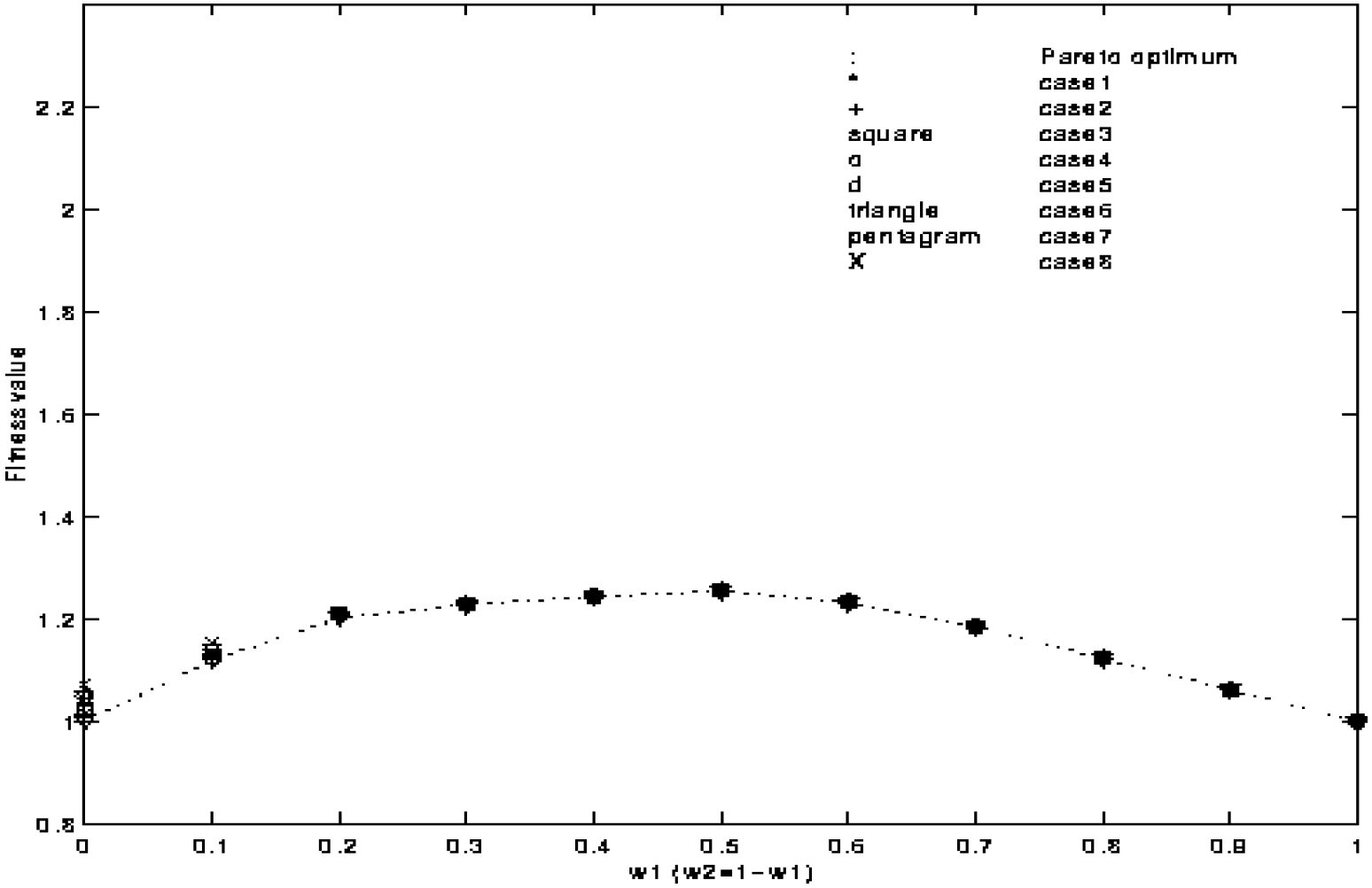
IN-Based Multicriterion Design

- Designate designs with best composite fitness as antigens - this is done for all chosen weight combinations
- Immune network simulation is used to promote convergence of subsets of populations to each antigen - specialist antibody production

Ten Bar Planar Truss

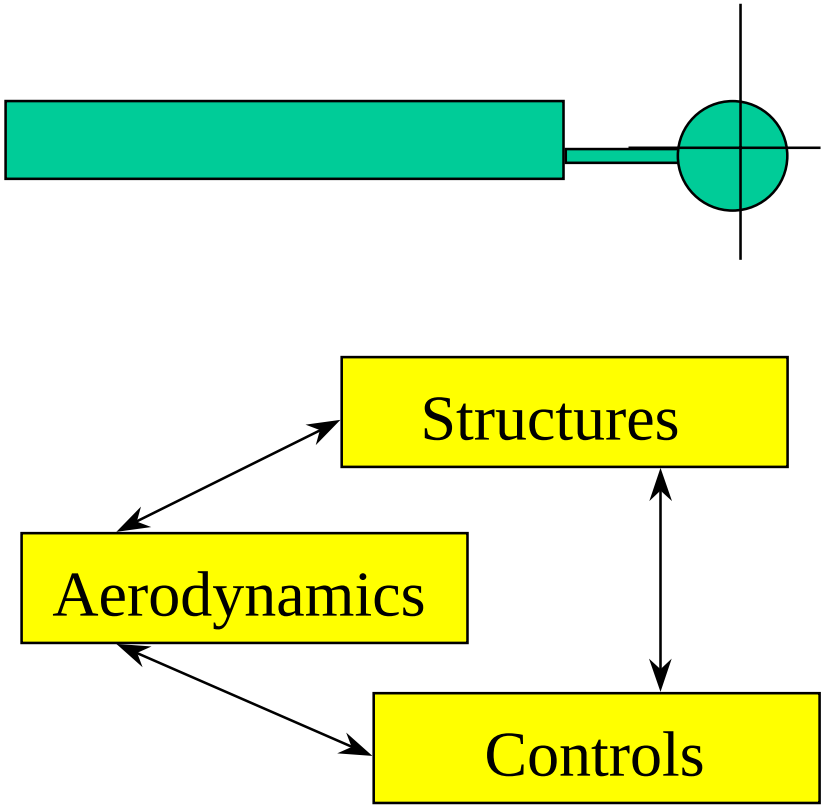


IN Based Multicriterion Design



Decomposition Based Design

- Must still contend with high cost of analysis
- How does one identify topology for decomposition
  - disciplinary?
  - maximum effectiveness?
- Account for subproblem interactions
  - gradients (not always possible)
  - non-gradient methods (GA)



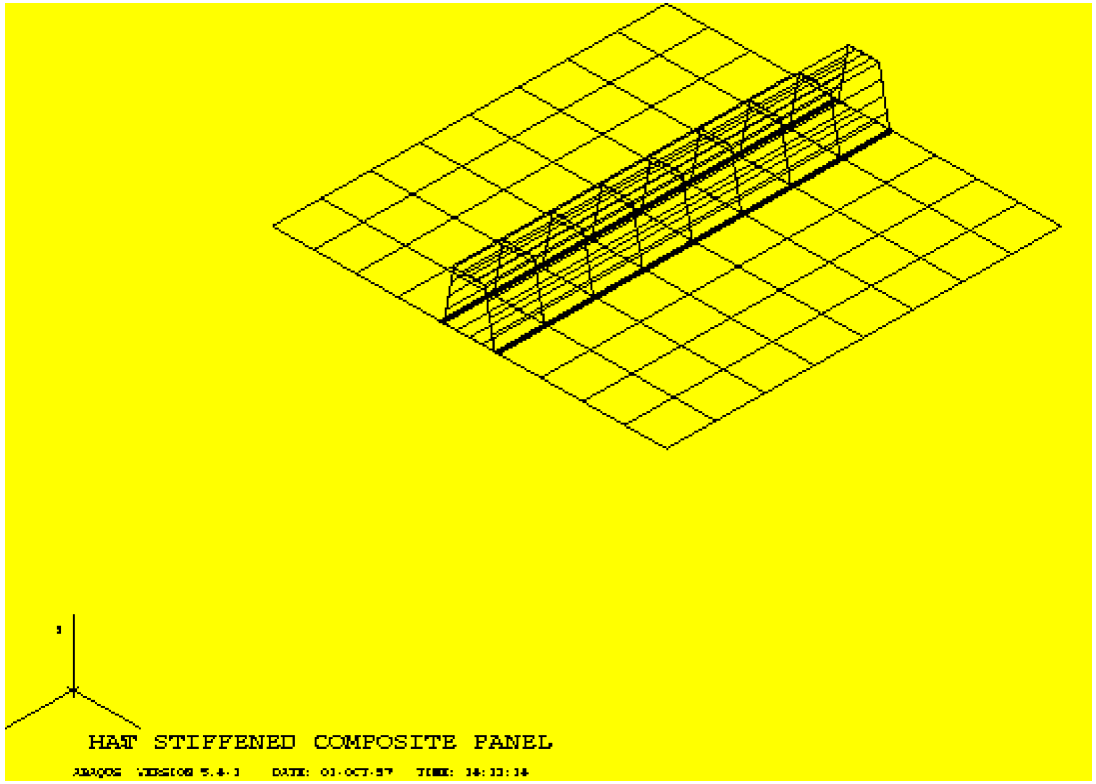
Approximations & Modeling

- Many analytical processes tend to be computationally demanding
- Creating models for manufacturing processes is difficult - example of tape lay-up process used to fabricate composite panels
  - models based on physical experiments
    - largely impractical
  - models derived from simulations
    - computationally and labor intensive
- Cost models

Approximations & Modeling

- Traditionally used Taylor series approximations are of limited value
  - global search
  - move limits
- Well chosen data sets to represent system behavior
  - uniform random distribution
  - design-of-experiments approach
  - machine learning or computational intelligence paradigms
- Approximations used for interpolation purposes - zero order
  - response surfaces
  - neural networks

Hat-Stiffened Composite Panel



Design Problem

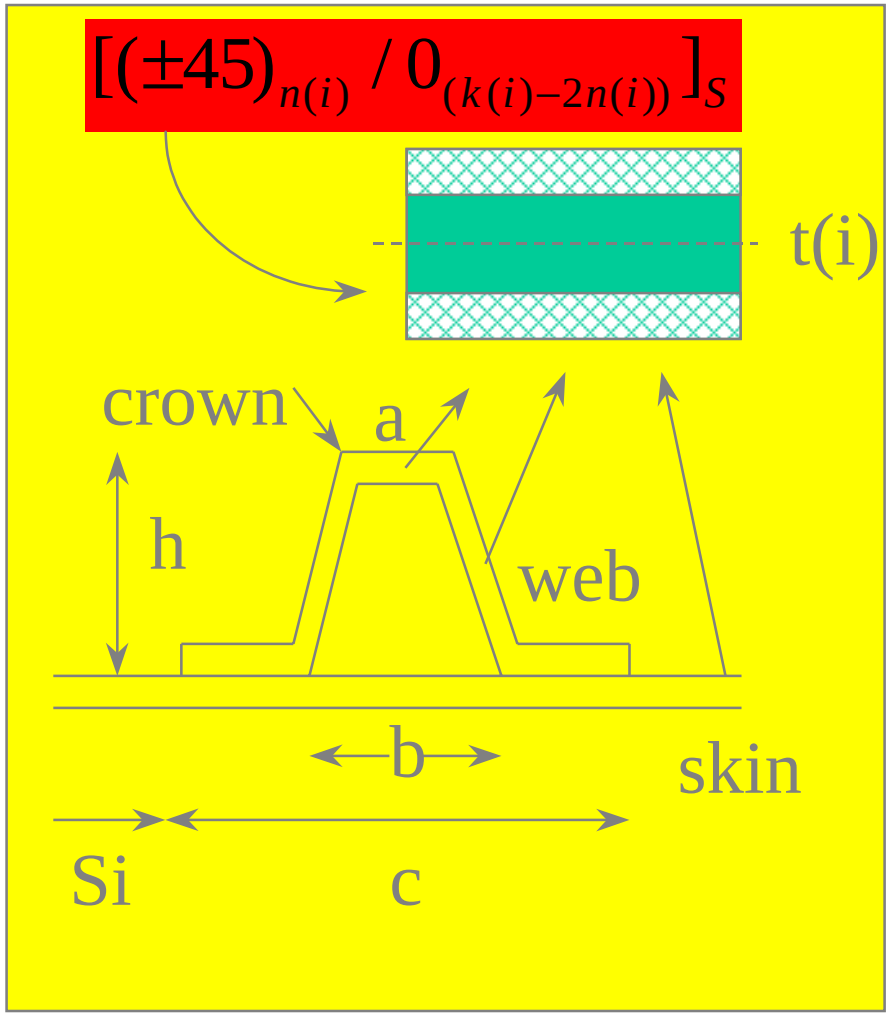
- Minimize $W = W_{plate} + \sum_{i=1}^{N} W_i$
- Subject to

$$g_1 \equiv \frac{P_{buck}}{P_{app}} - 1 \le 0 \qquad g_2 \equiv \frac{\sigma_{max}}{\sigma_{all}} - 1 \le 0$$

$$g_3 \equiv \frac{d_{max}}{d_{all}} - 1 \le 0$$

$$g_4 \equiv \frac{\omega}{n\Omega - \delta} - 1 \le 0$$

$$g_5 \equiv 1 - \frac{\omega}{n\Omega + \delta} \le 0$$



Design Variable Description

Table 1: Design variables in neural network training

design variable		lower	upper	Δ	type
S_1	spacing	0.0	14.0	0.1	discrete
c	floor	4.0	7.8	0.1	continuous
b	web	2.0	6.0	0.1	continuous
a	crown	0.5	4.0	0.1	continuous
h	height	1.5	5.0	0.1	continuous
n_1	$\pm$45 layups in skin	0	8	1	integer
n_2	$\pm$45 layups in web	0	6	1	integer
n_3	$\pm$45 layups in crown	0	6	1	integer

BP Network Approximation

Table 2: Performance of BPN network for function approximation

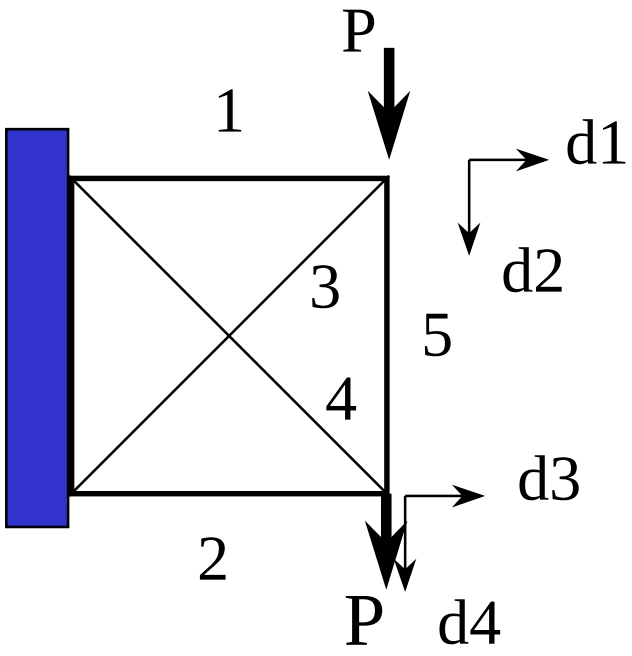
natural frequency (ω)				buckling load ($P_{buckling}$)		
actual	NN	error (%)		actual	NN	error (%)
45.848	46.064	0.47		1746.2	1783.5	2.13
45.035	44.015	2.26		1607.6	1690.7	5.17
44.330	44.577	0.55		1690.7	1720.5	1.75
47.015	46.836	0.37		1797.1	1773.0	1.34
43.611	43.614	0.01		1862.1	1835.2	1.44
51.306	51.326	0.03		1887.4	1803.4	4.44
30.630	30.261	1.20		1783.9	1829.0	2.52
32.097	32.000	0.30		1815.3	1768.7	2.56
33.292	33.690	1.19		1556.6	1546.2	0.66
46.746	46.685	0.13		1641.0	1717.5	4.66

BP Network Approximation

maximum deflection (d_{max})				maximum von Mises stress (σ_{max})		
actual	NN	error (%)		actual	NN	error (%)
0.0410	0.0422	2.96		53570	54110	1.00
0.0714	0.0711	0.30		43230	42610	1.43
0.1041	0.1080	3.71		40080	42010	4.81
0.0658	0.0644	0.92		49880	48780	2.19
0.0481	0.0475	1.12		87390	91030	4.17
0.0975	0.0978	0.42		95110	100900	6.15
0.0854	0.0841	1.55		109500	114700	4.75
0.0368	0.0389	5.56		244600	257200	5.15
0.1317	0.1378	4.66		62670	65860	5.09
0.0555	0.0575	3.51		42510	43730	2.86

Extracting Causality From Data

Neural network mapping generated by using 50 randomly generated data sets - backpropagation network



	A1	A2	A3	A4	A5
d1	1.0000	0.1490	-0.315	0.3980	-0.0030
d2	0.3500	0.3290	1.0000	0.9730	-0.0130
d3	-0.144	-1.0000	0.3510	0.4050	0.0080
d4	0.3500	0.3140	0.9200	1.0000	-0.0130

Causality from Trained BPN

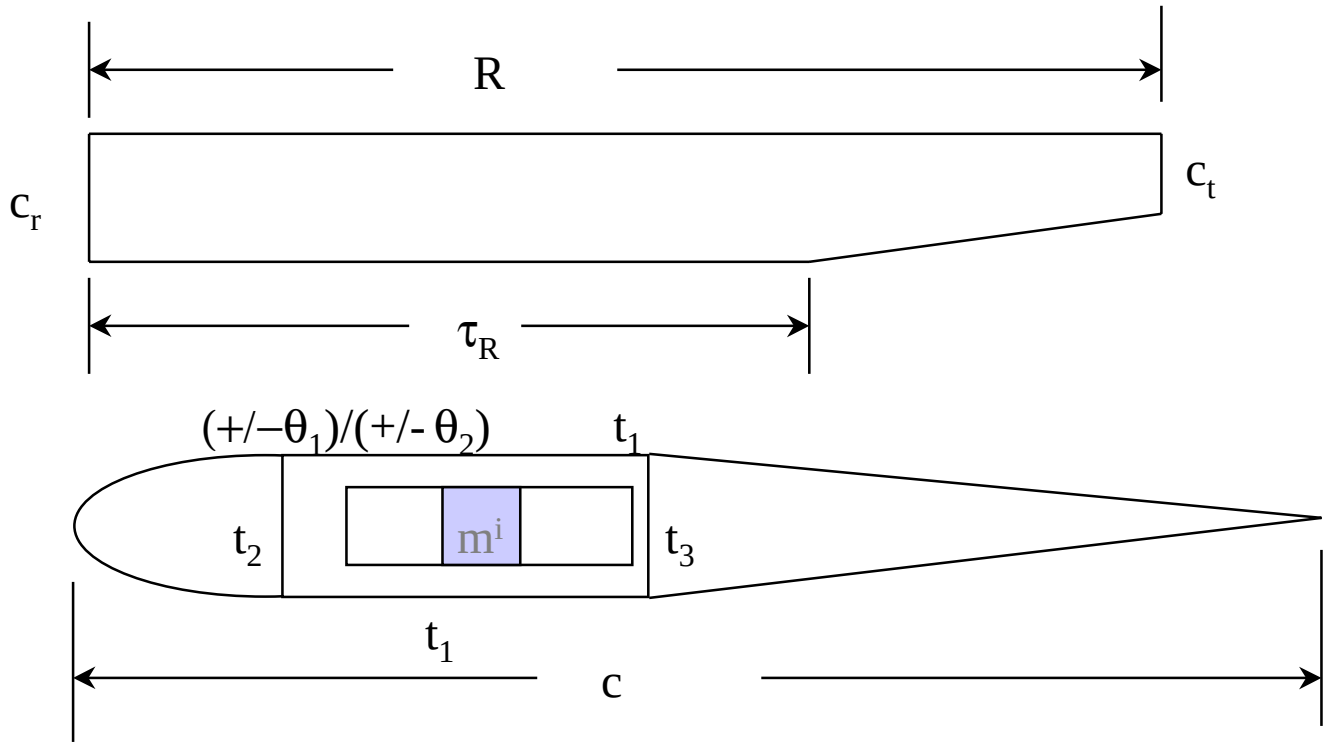
- Compute first

$$T_{ij} = [W]_{n_o \times n_1} \times [W]_{n_1 \times n_2} \times \cdots \times [W]_{n_{N-1} \times n_N}$$

  - where n_i is the number of neurons in the ith layer, i=0,N, and $[W]_{n_i \times n_{i+1}}$ is a matrix of interconnection weights between layers i and i+1
  - These coefficients are normalized as follows

$$R_{ij} = \left[\frac{T_{ij}}{\max_i |T_{ij}|} \right] \quad i = 1, n_o, \; j = 1, n_N$$

Decomposition - Blade Design



Mixed Variable Space

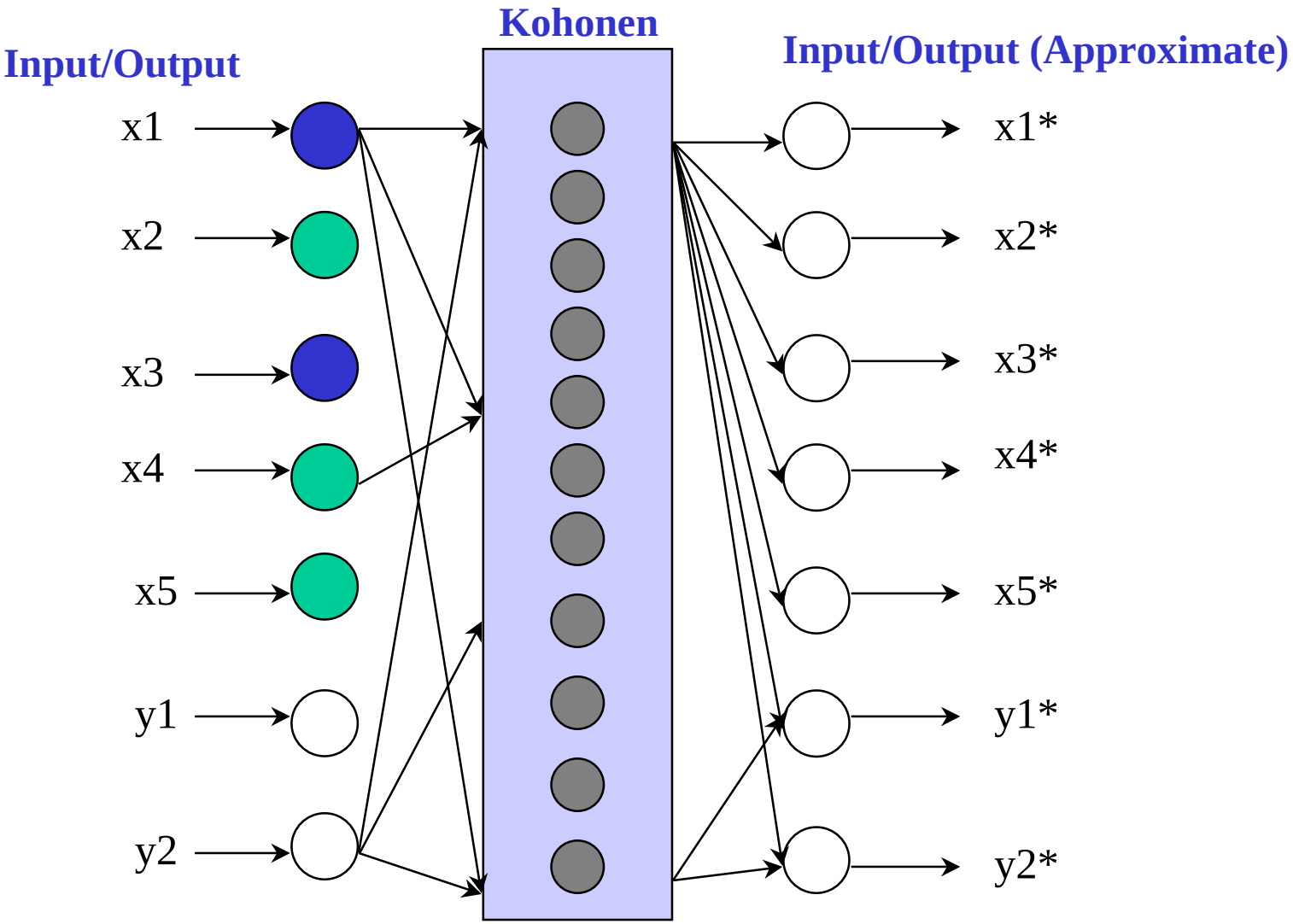
Design Variable	*ID*	*Type*
horizontal flange thickness	t_1^i	continuous
tuning mass	m^i	continuous
vertical web thickness	t_2^i, t_3^i	continuous
blade twist	θ_t	continuous
twist shape parameter	δ	continuous
taper inception point	τ_R	continuous
chord ratio	λ_c	continuous
rotational speed	Ω	integer
composite layup angle	θ_1,θ_2	discrete

where, i denotes the blade spanwise segment

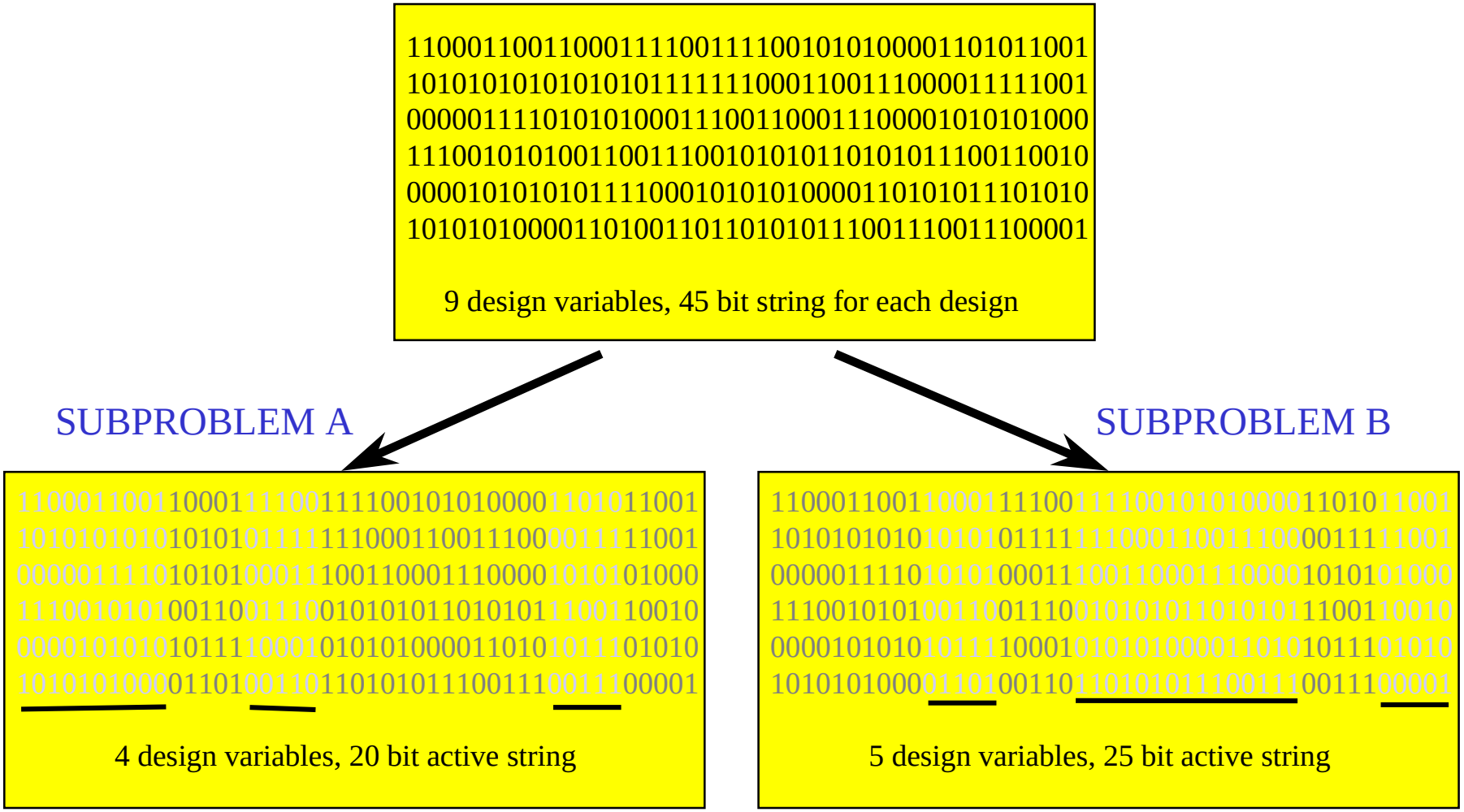
Causality Based Partitioning

	Subsystem A	Subsystem B	Subsystem C
Objective Function	$F(X) = c_1 F_z + c_2 M_y + c_3 M_z$		
Constraints	$HP_h \le HP_a$ $\eta^L \le \eta \le \eta^U$ $AI \ge AI^L$	$HP_f \le HP_a$ $(c_T/\sigma)^L \le c_T/\sigma \le (c_T/\sigma)^U$	$W_b \le W_b^U$ $\overline{R} \le 1$ $\sigma_{buck} \le \sigma_{all}$
Design Variables	m_1, m_2, m_3, t_1^4 $t_1^5, t_1^6, t_2^3, t_2^6$ $t_3^6, t_3^8, t_3^9, t_2^9$ $\tau, \pm\theta_1$	$t_1^3, t_1^{10}, t_2^4, t_2^5, t_2^7, t_2^8, t_3^2$ $t_3^3, t_3^5, t_3^7, t_3^{10}, \theta_t, \delta, \pm\theta_2$	m_4, m_5, t_1^1, t_1^2 $t_1^7, t_1^8, t_1^9, t_2^1$ $t_2^2, t_2^{10}, t_3^1, t_3^4$ λ_c, Ω

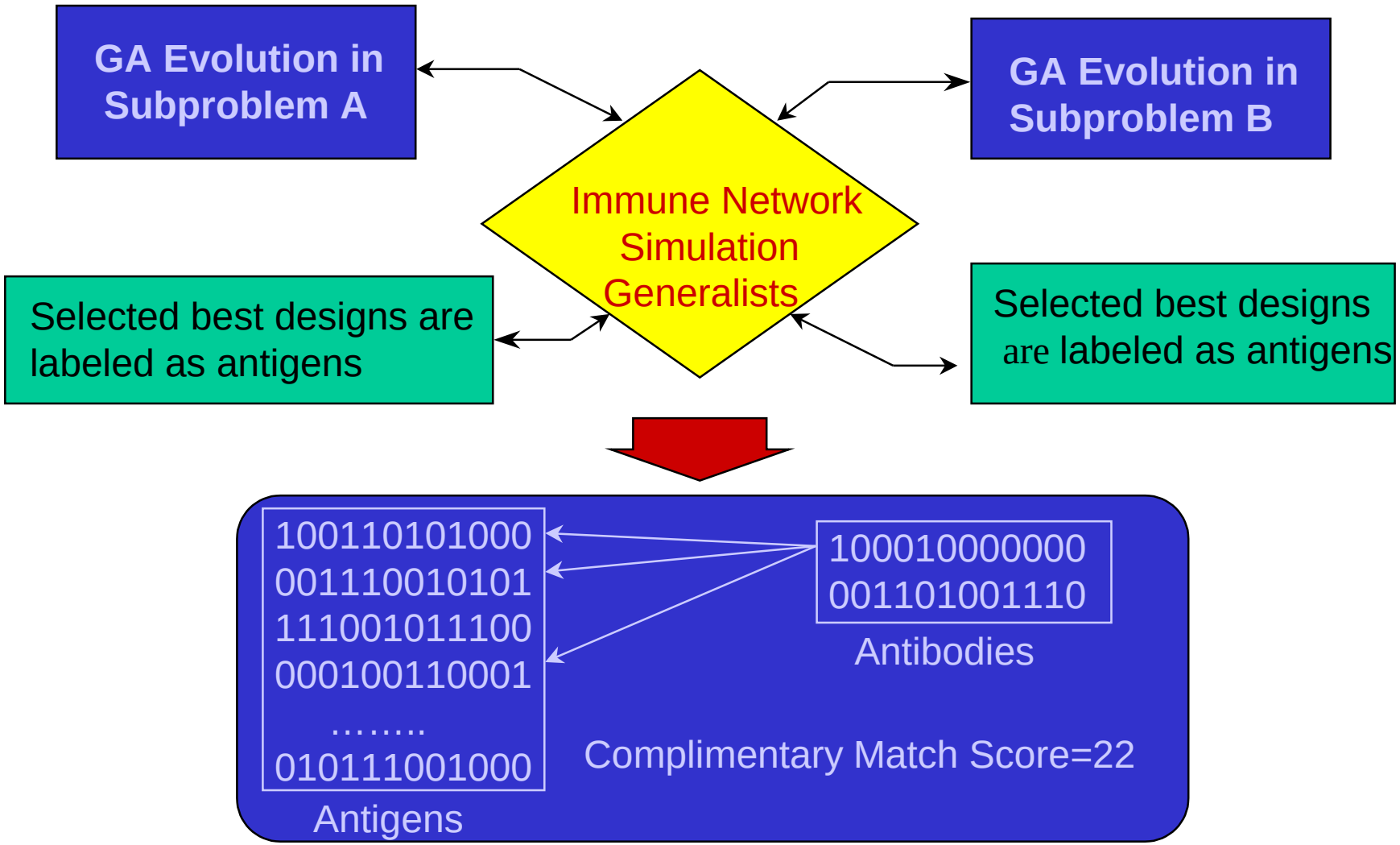
Problem Co-ordination - CP Network



Problem Co-ordination
Immune Network Modeling



Co-ordination : IN Simulation



Computational Intelligence

- Principal motivation here is to overcome pitfalls of static rule bases
  - tightly focused problems
  - narrow domain of application
- Rules are evolved on the basis of their performance in attaining goals
- New rules better configured to improving problem solution emerge as an outcome
  - both the strength and context of a rule is subject to evolution

Computational Intelligence
A Simple Illustration

- Requirement to size truss structure for minimal weight and stress constraints
- Fully stressed design => member overstress then increase element area; conversely, member understress, then decrease element size
- Code this rule as a binary string, and create a population of such rules

1001	-	011 100 111 010 111 011	(0.65)
Condition		Actions	Strength

- Examine effectiveness of rule in accomplishing goal - reward or penalize fitness - evolve population of rules using GA

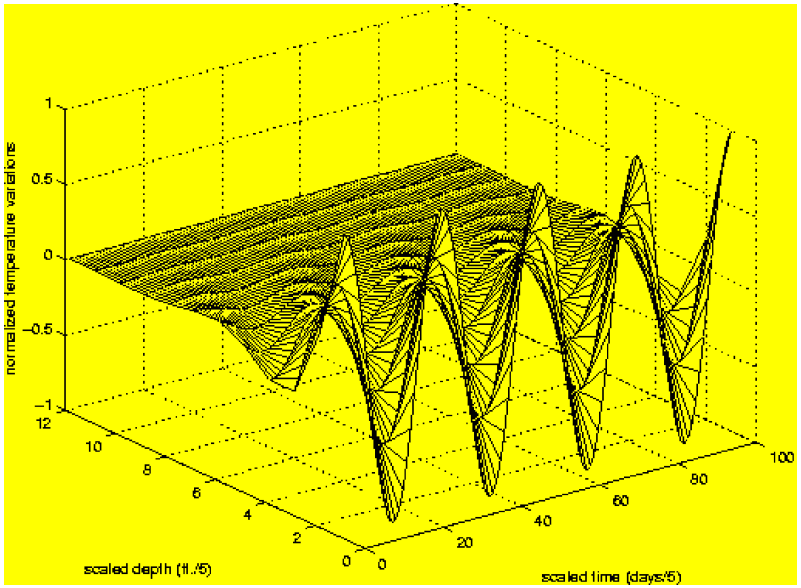
Improved Response Surfaces

- Determine the number of training exemplars required, and to identify the region in which these must be distributed
- Process initiated with
  - some initial distribution of data
  - assumed order of polynomial
  - test patterns not used in training but for which exact output is known
- Define classifiers randomly - then evolve classifiers
  - condition - location of pattern and associated approximation error
  - action - number/distribution of new training exemplars

Numerical Examples

- Temporal variation in temperature of earth's crust

$$\delta = \cos(0.05x_1 - 0.2x_2)\exp(-0.1x_2)$$

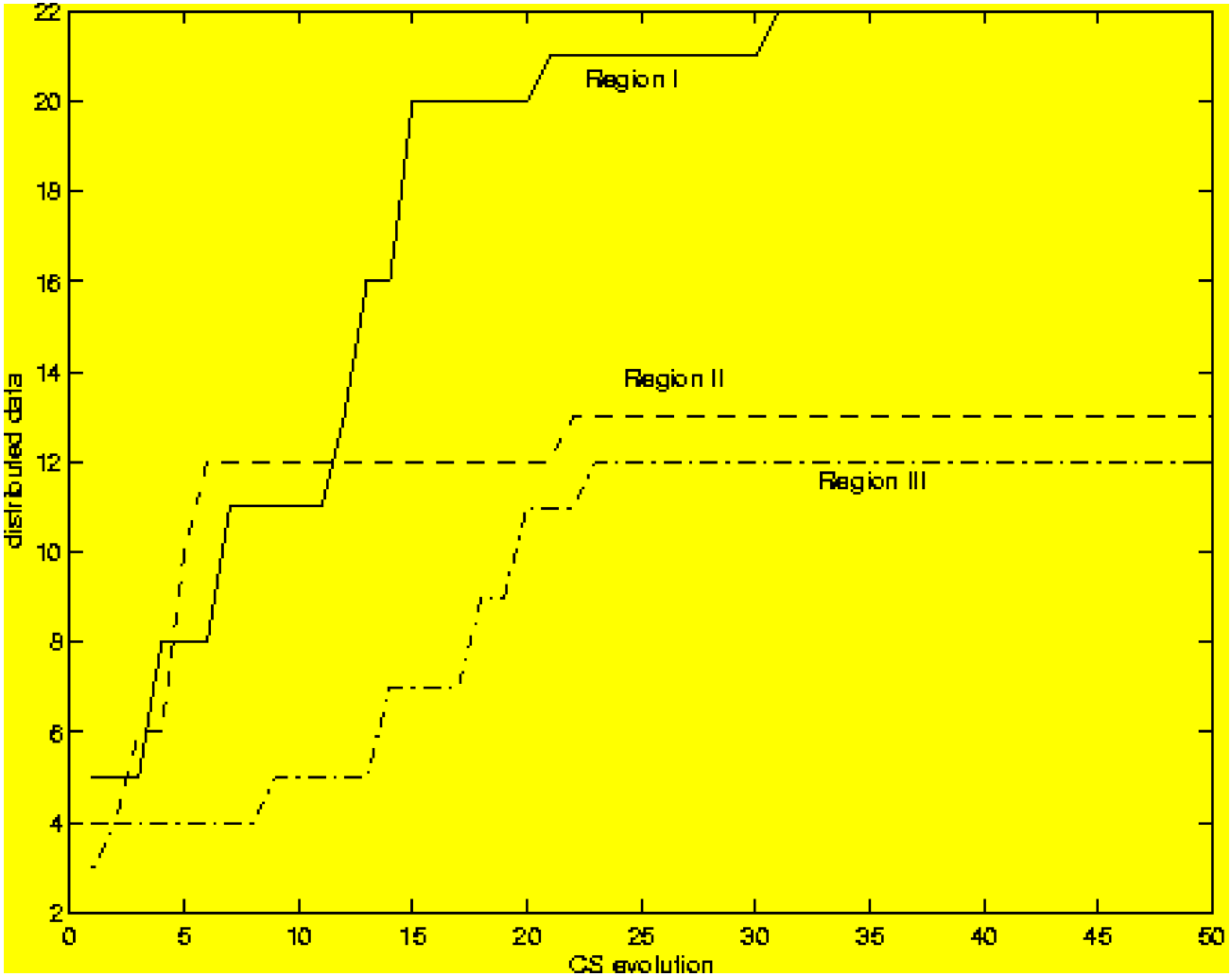


Region I - 0 ft to 15 ft
Region II - 15 ft to 50 ft
Region III - 50 ft to 60 ft

Results - Algebraic Function

Network Training	Generalization Error			Training Error
	Region I	Region II	Region III	
100 uniformly distributed patterns	0.5982	0.0299	0.0421	0.08234
50 uniformly distributed patterns	0.6763	0.0379	0.0476	0.07231
47 classifier distributed patterns	0.0890	0.0260	0.0008	0.04394

Results - Algebraic Function



Results - Algebraic Function