GUIDED RANDOM SEARCHES

Optimization

- Exhaustive search
- Random search
- Greedy search
- Evolutionary search

Evolutionary practitioners

- Believe in nature

- Nature finds great solutions only with three tools
  - Survival of the fittest
  - Mating
  - Mutation

What does survival of the fittest mean

CSE 7303c

Summary of biology

- Not all members survive until reproduction. Only a few learn to earn lunch. Others are lunch!

- The strong ones then need to attract and mate other members of the species that survived.

- The offspring generation hence is normally better than the parents generation

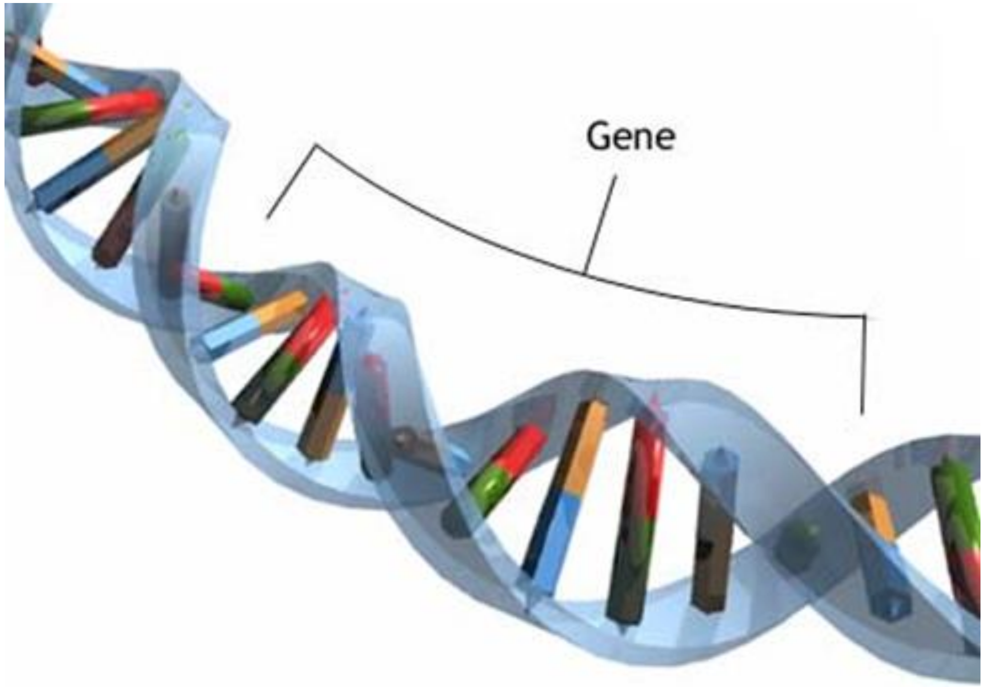
- Over generations, species develop

Can a similar theory be applied to optimization

- Start with random solutions
- Kill the weak ones
- Reproduce from the good ones; Mutate infrequently
- Will the newer set be more optimal than its parents?

Gene

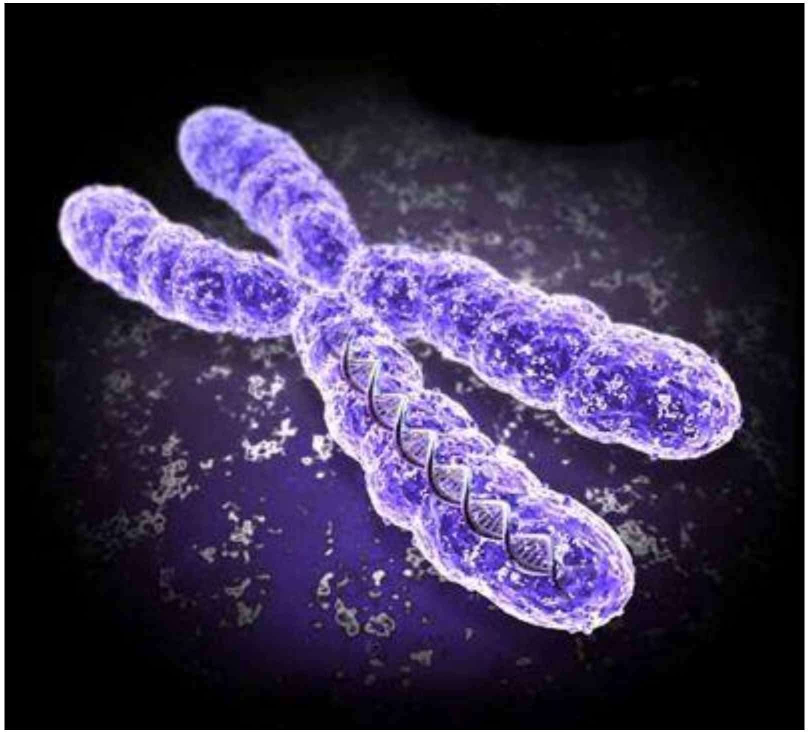
Gene: The reproducible building block of chromosome



CSE 7303c

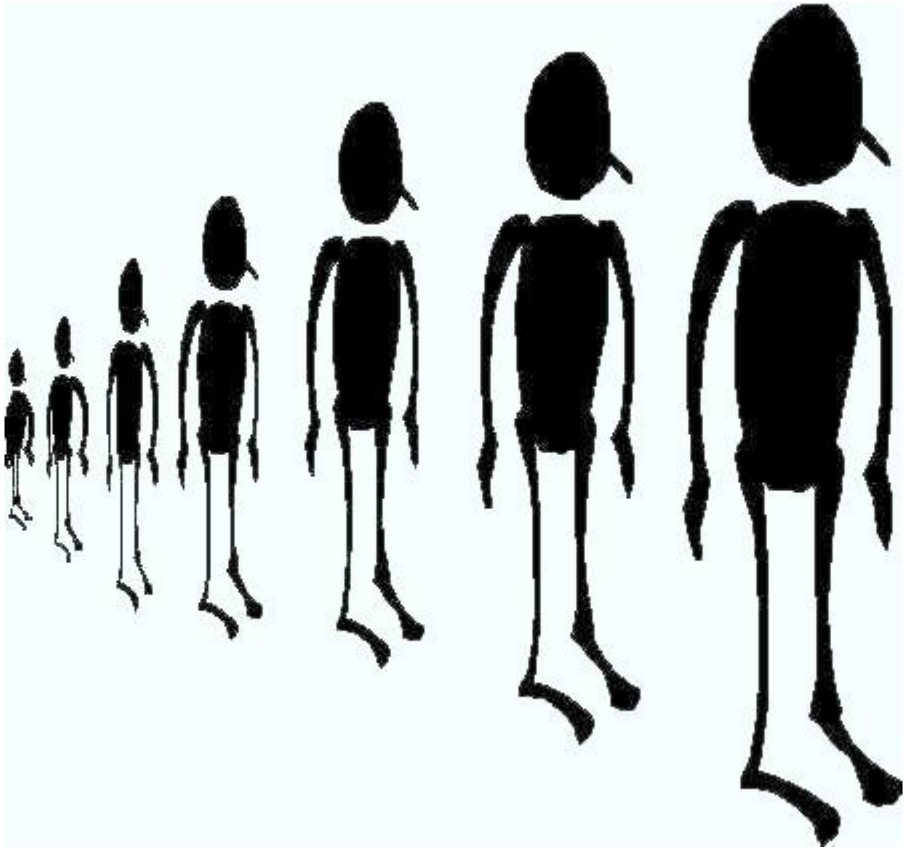
Beg, Borrow or Steel

- Chromosome: A possible solution



Population

- A set of chromosomes



Generation

- A population derived from the fittest chromosomes of the previous population

Reproduction, Mating and Mutation

- Create a new solution from the old fitter solutions

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

So, GA at a snapshot

- Create a schema to create possible solutions in a bit/byte format
- Start with a number of random solutions
- Identify the best ones
- Create new ones from them using some exchange mechanism
- Continue with the last two steps until a solution is found

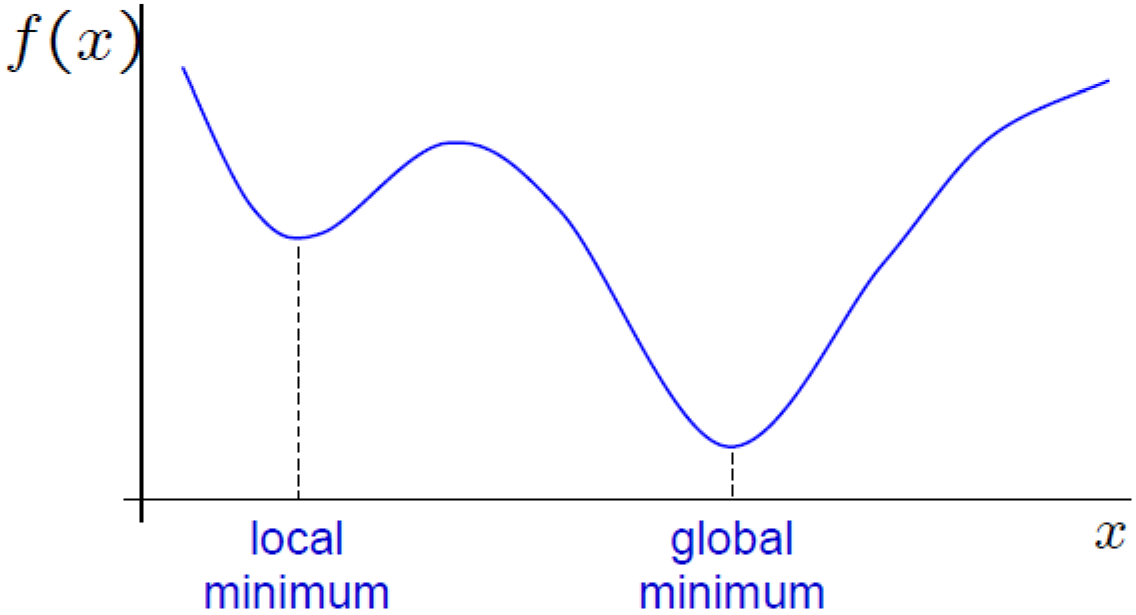
Some unique differentiators

- GAs work with a coding of the parameters and not the parameters themselves
- GAs search from a population of points and not a single point
- GAs work with the objective function and not the derivatives
- GAs transition based on random rules and not on deterministic rules

Optimizing non-convex functions

function of one variable

$$\min_x f(x)$$



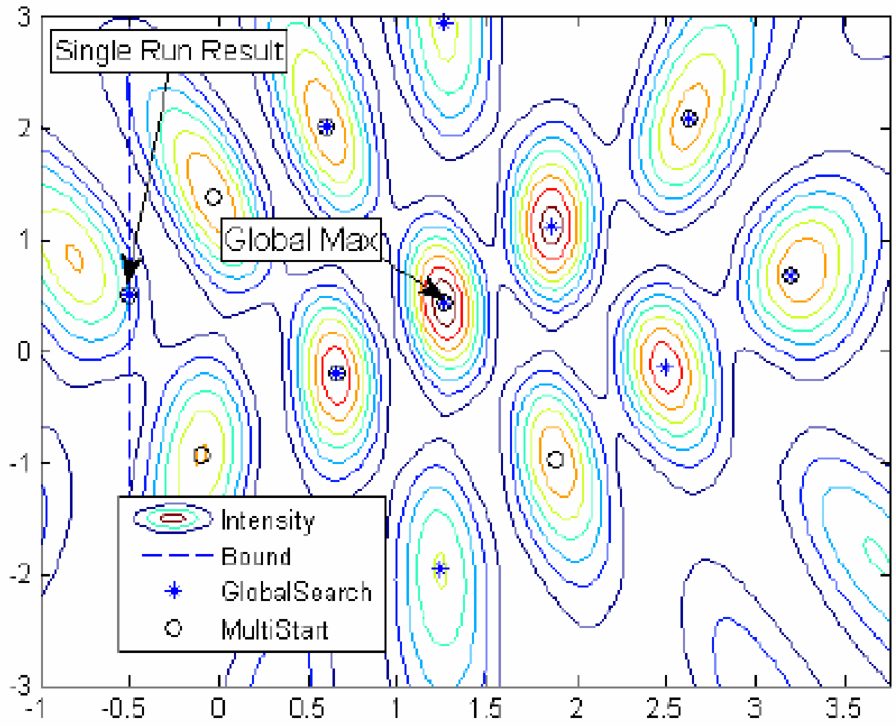
Sketch three methods:

1. grid search: uniform grid space covering
2. multiple coverings: Newton like methods within regions
3. simulated annealing: stochastic optimization

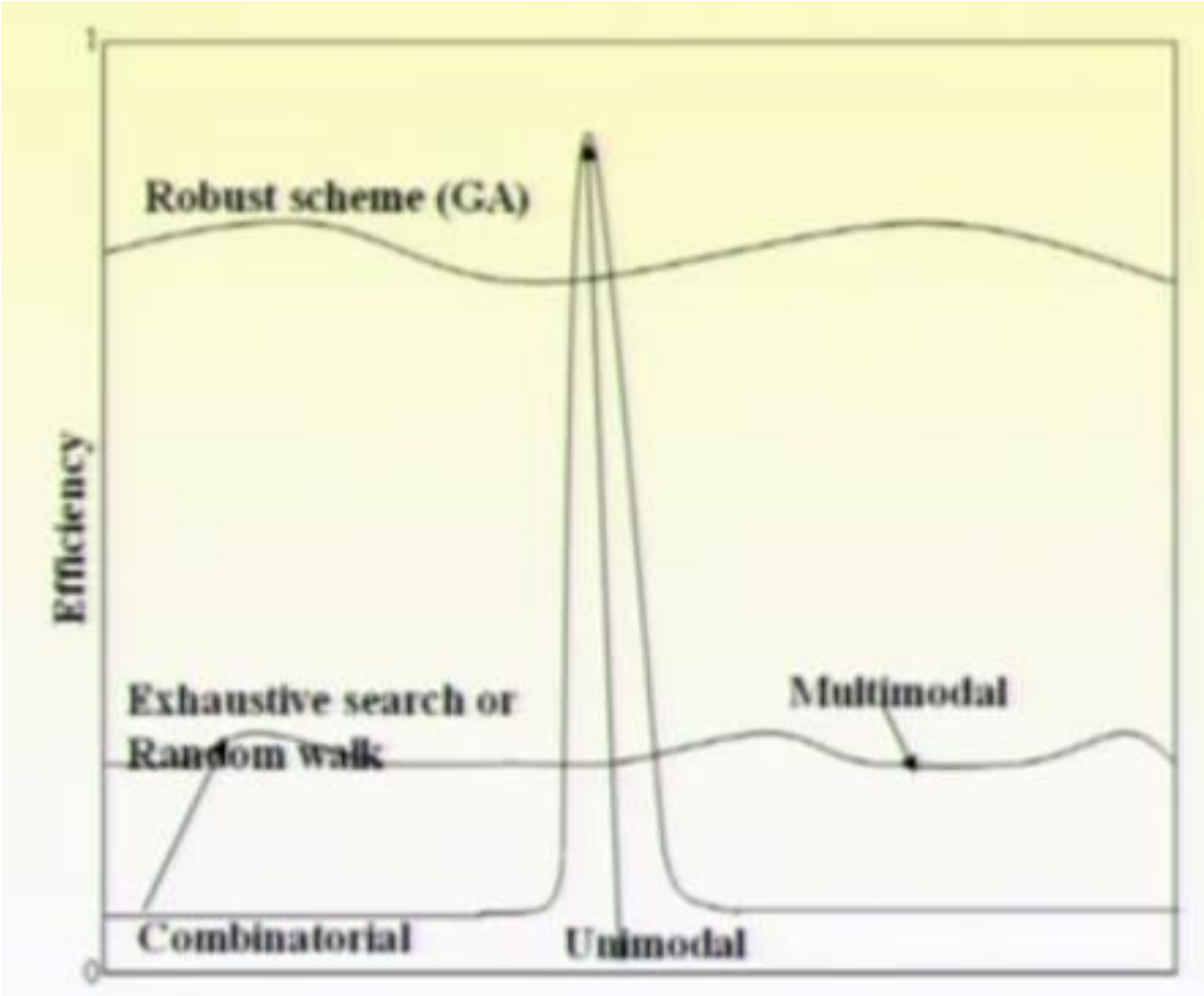
CSE 7303c

Multiple coverings ctd

- Use multiple starting points
- Continuous optimization method for each
- Record optimum for each starting point
- Sort values to find global optimum



CSE 7303c



1
0
Efficiency
Robust scheme (GA)
Exhaustive search or
Random walk
Multimodal
Combinatorial
Unimodal

CSE 7303c

Why do they work?

- Understanding the schema
  - What is a schema
    - 1, 0 or either
    - Length: Distance between two fixed points
    - Order: Number of fixed points
    - Average fitness

- Which schema are likely to survive under reproduction

- Fundamental law of genetic algorithms

  - Short, low order and above average schema grow exponentially

OPERATIONAL ASPECTS

Population

Chromosomes could be:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Binary is better

Binary	Octal	Fitness
000	0	22
011	3	8
101	5	11
111	7	3

Binary requires less coding effort

Innovations: Creating a coding scheme

- Password retrieval
- Knap Sack
- Polynomial fit
- TSP
- Portfolio allocation

How do we decide mating pool

Figure 2. Selection application

How do we determine reproduction

- Most common: Cross over
  - npoint crossover: n crossover points are randomly selected and the segments of the parents, defined by them, are exchanged for generating the offspring.
  - uniform crossover: the values of each gene in the offspring are determined by the uniform random choice of the values of this gene in the parents.

For real coding

Arithmetical crossover (Michalewicz, 1992)

Two offspring, $H_k = (h_1^k, \ldots, h_i^k, \ldots, h_n^k)$ $k = 1, 2$, are generated, where $h_i^1 = \lambda c_i^1 + (1 - \lambda)c_i^2$ and $h_i^2 = \lambda c_i^2 + (1 - \lambda)c_i^1$. λ is a constant (uniform arithmetical crossover) or varies with regard to the number of generations made (non-uniform arithmetical crossover).

CSE 7303c

- Flat crossover: An offspring, H is generated, where hi is a randomly (uniformly) chosen value of the interval [c1i; c2i]

CSE 7303c

Cross over

- Hello World
- Knap Sack
- Polynomial fit
- TSP
- Portfolio allocation

Inversion or Mutation

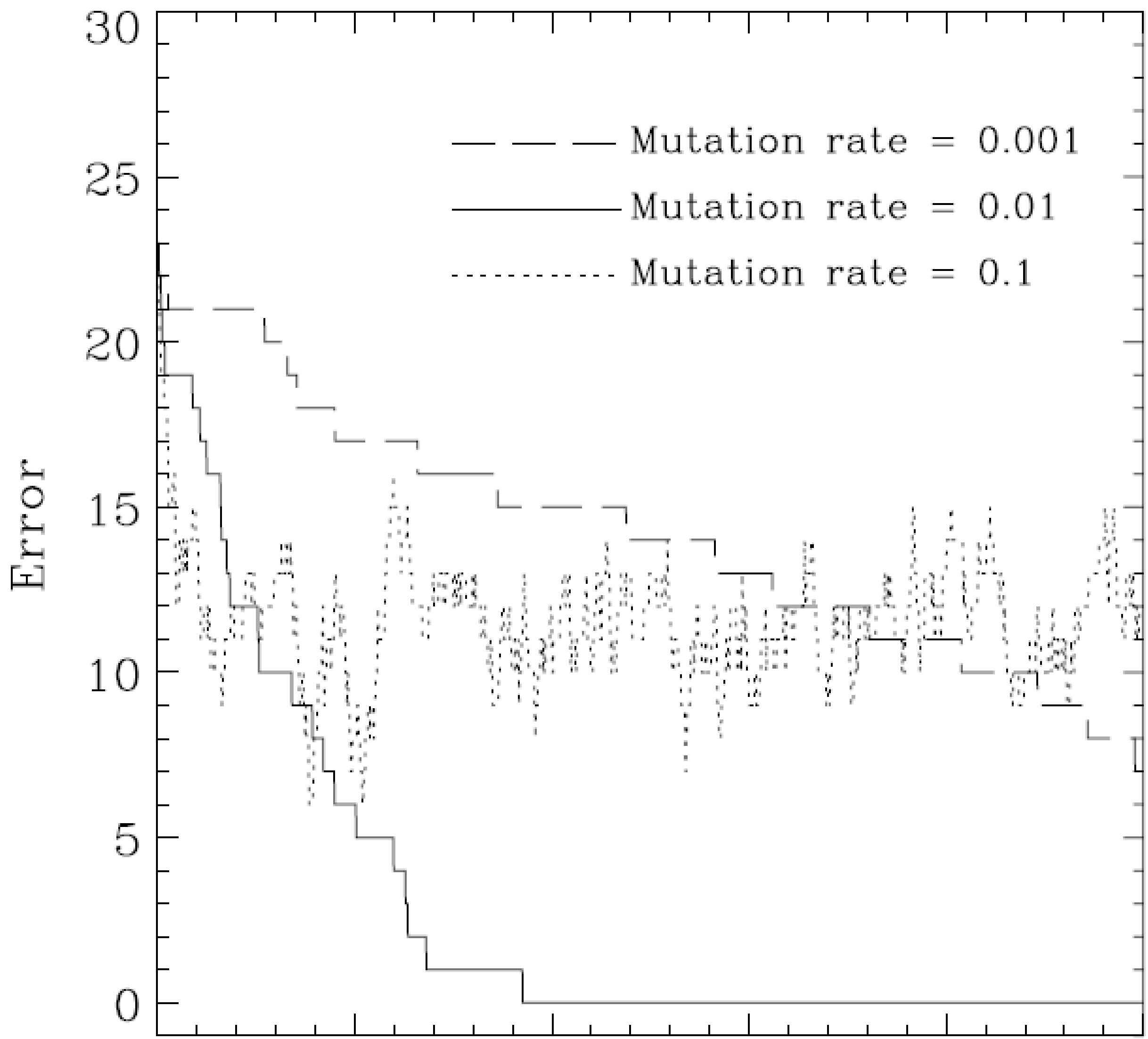
- Flip
- Random generation
- An unexpected operation

Non uniform mutation

- Initially mutate heavily

- As the number of generations increase mutate less

What about mutation?

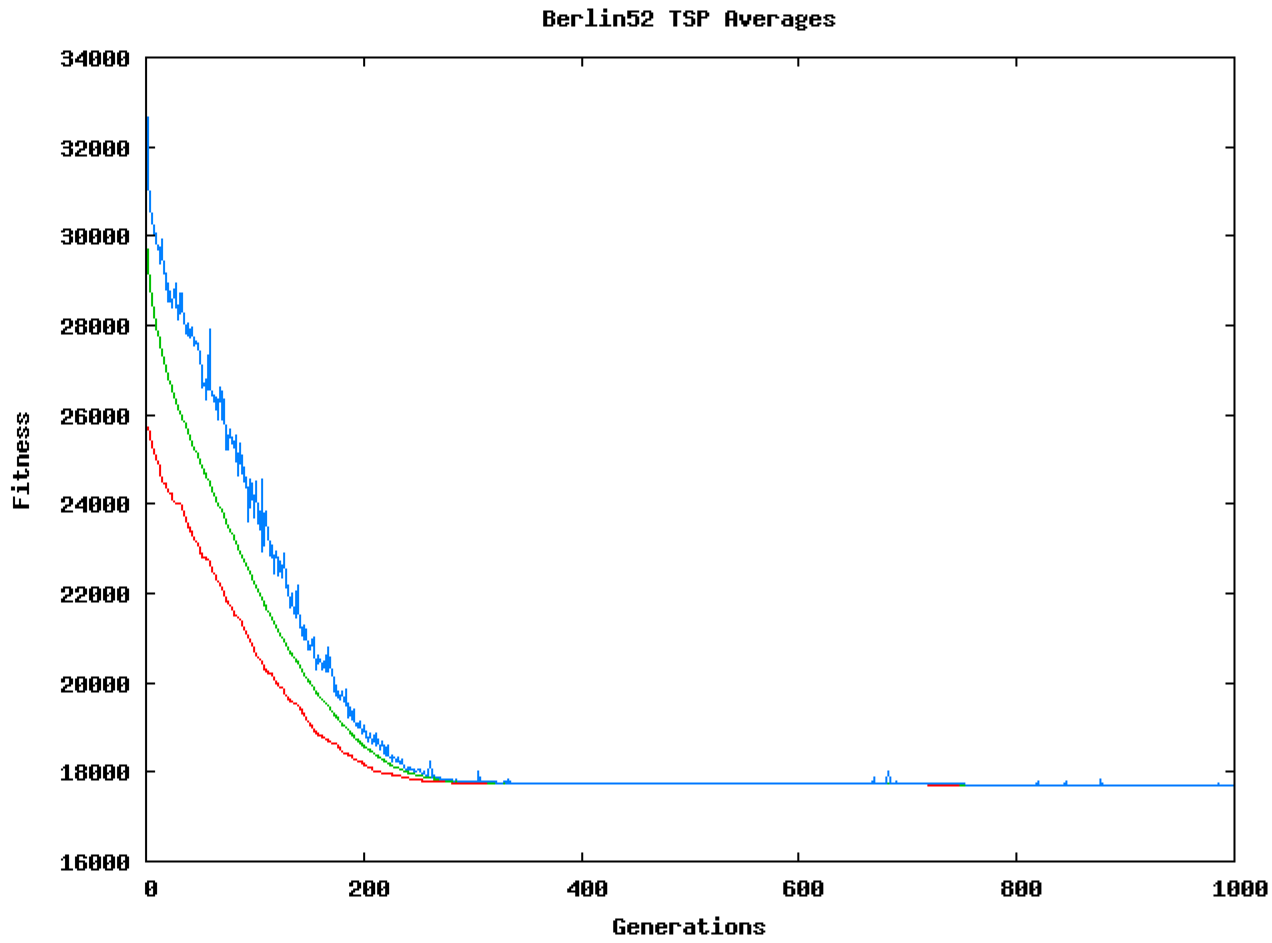


Mutation scheme

- Hello World
- Knap Sack
- Polynomial fit
- TSP
- Portfolio allocation

Convergence of a GA

When to stop?



EXAMPLES

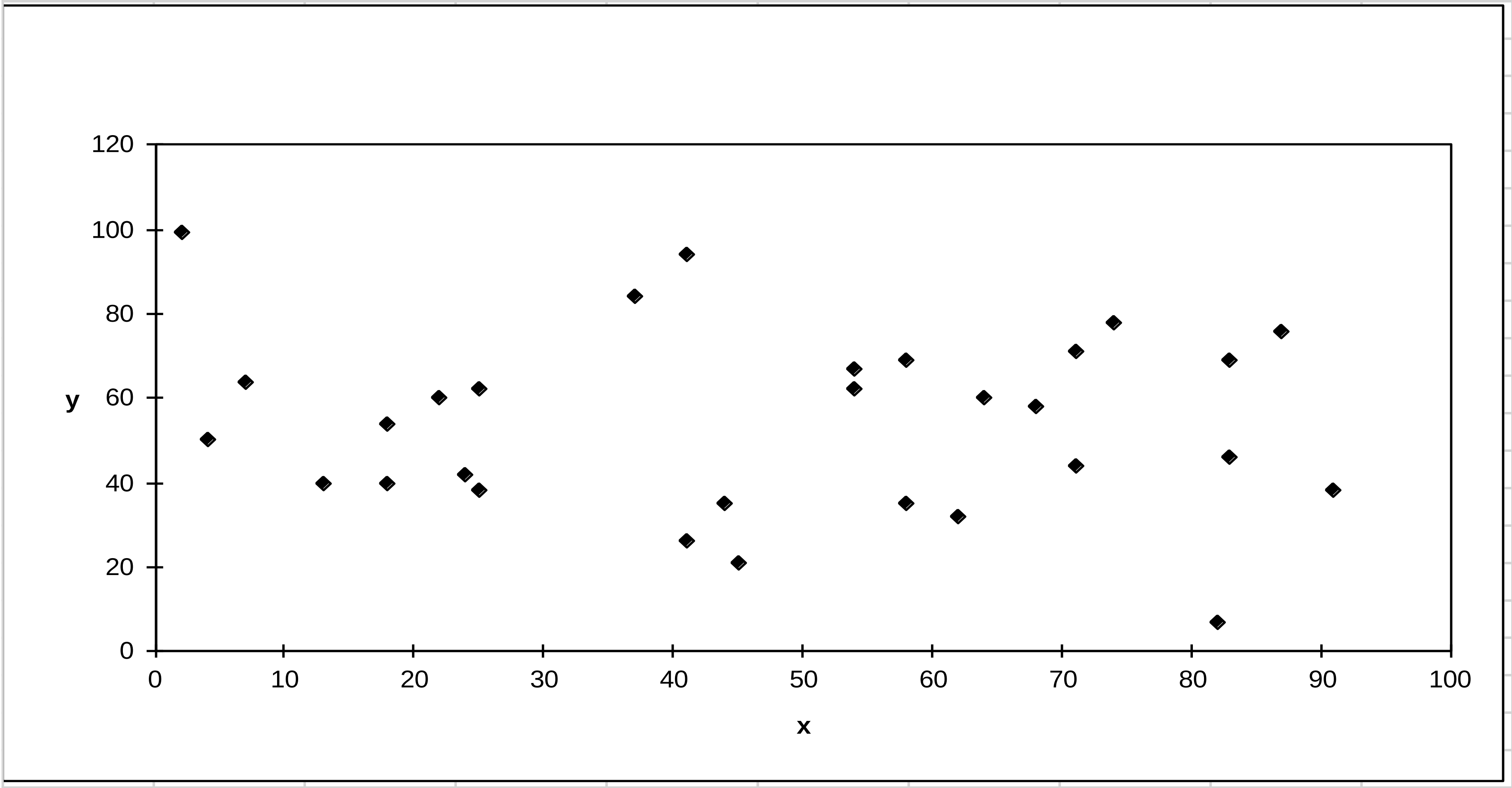
A Simple Example

The Traveling Salesman Problem:

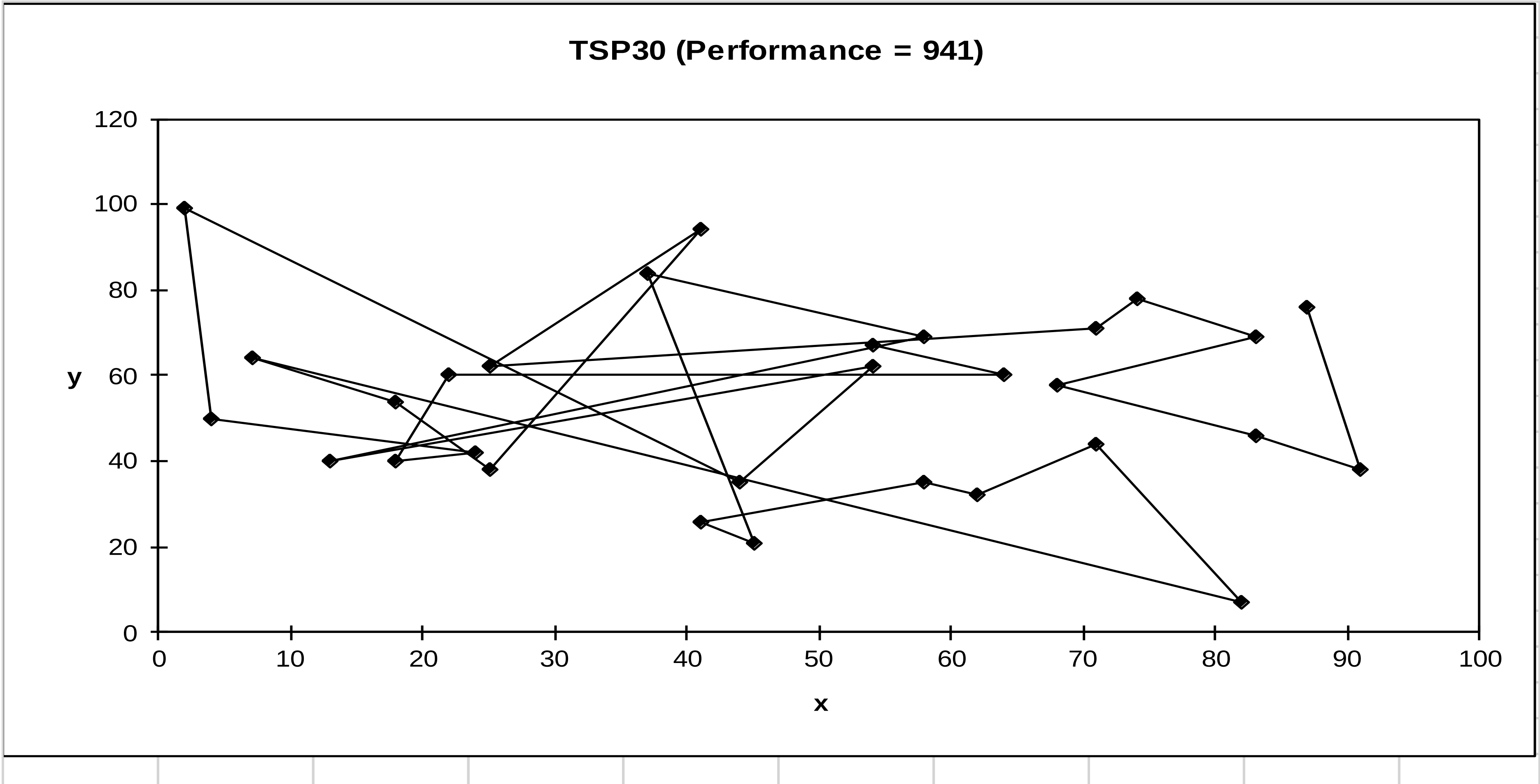
Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

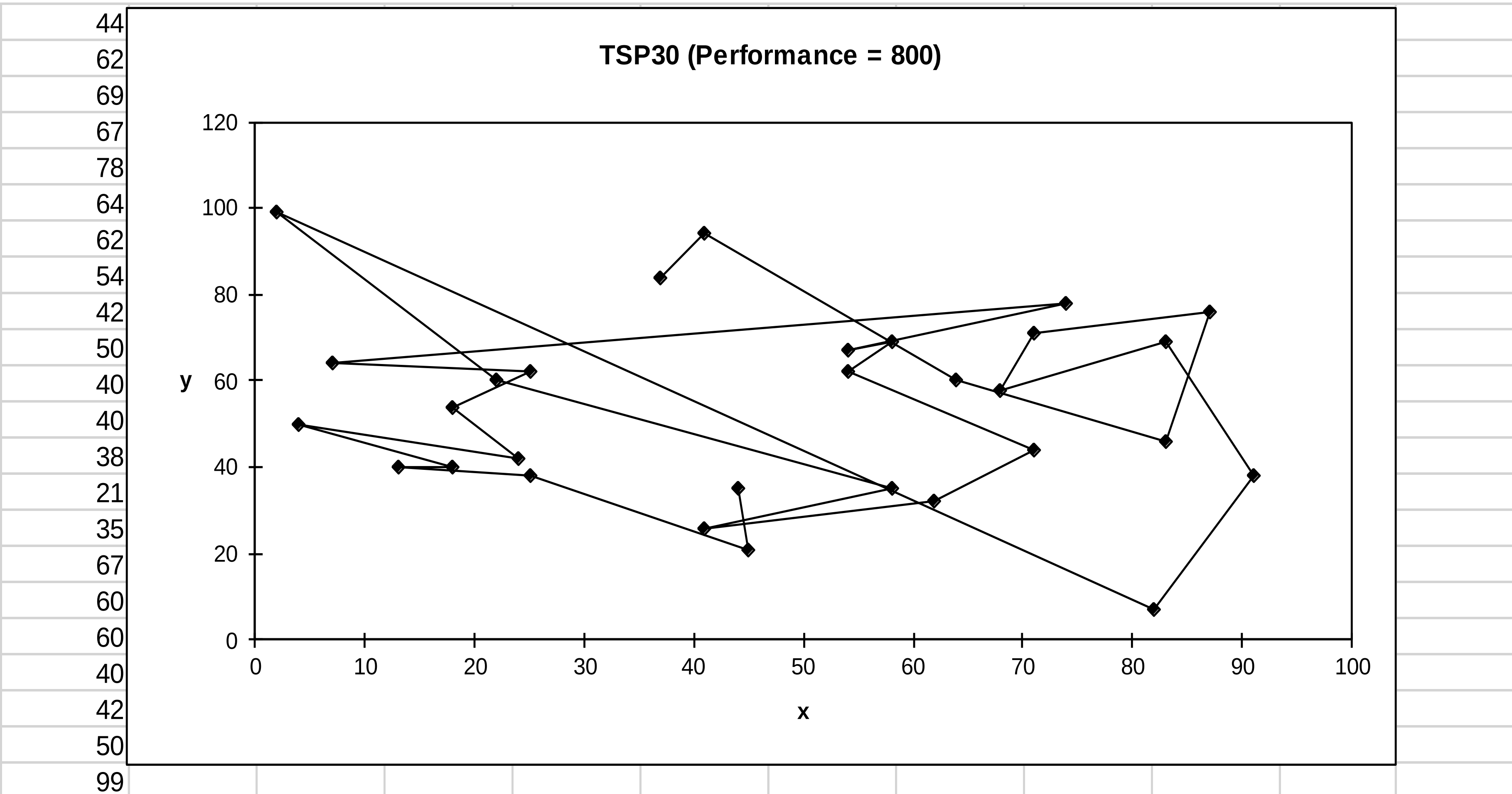
TSP Example: 30 Cities



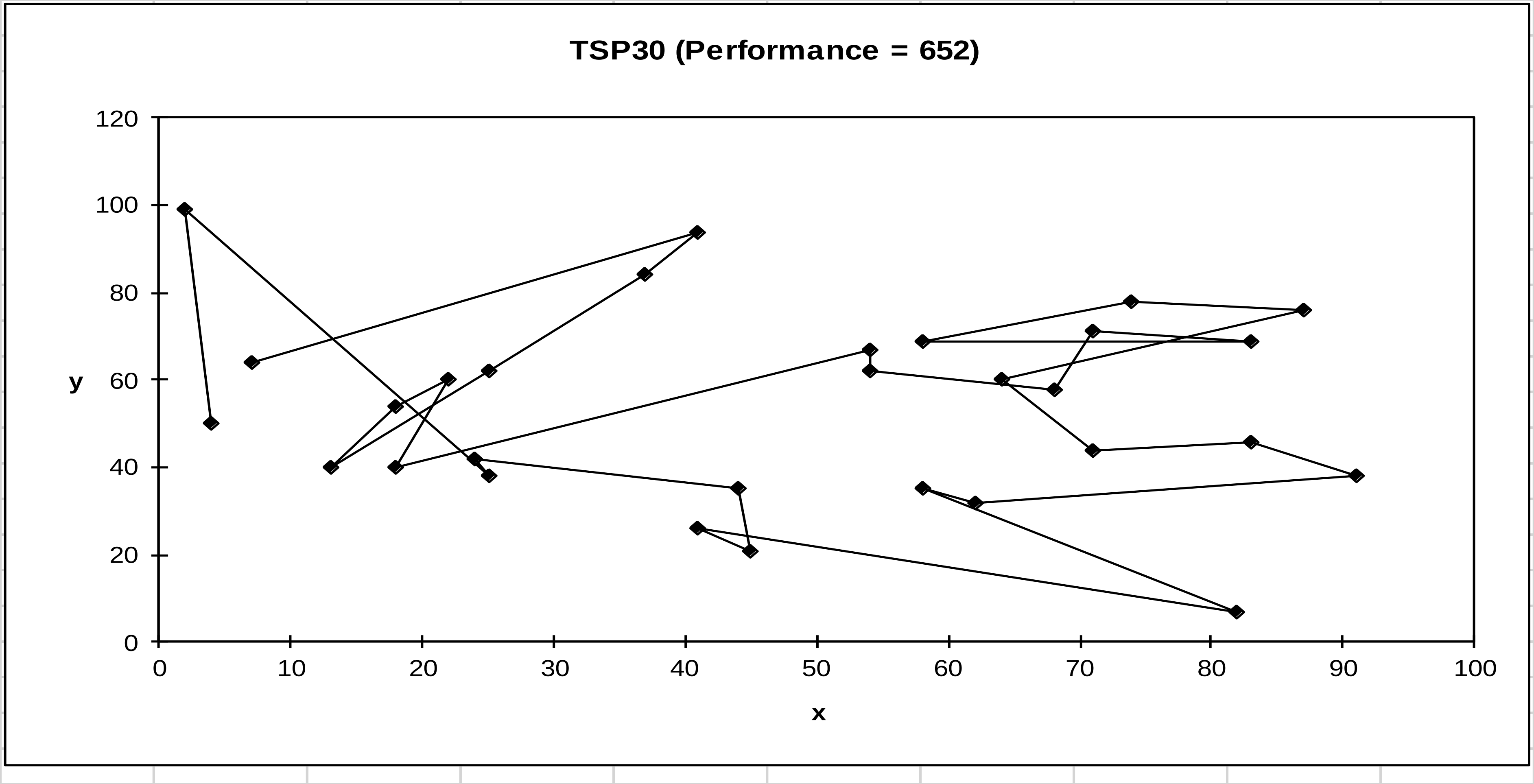
Solution $_i$ (Distance = 941)



Solution $_j$(Distance = 800)

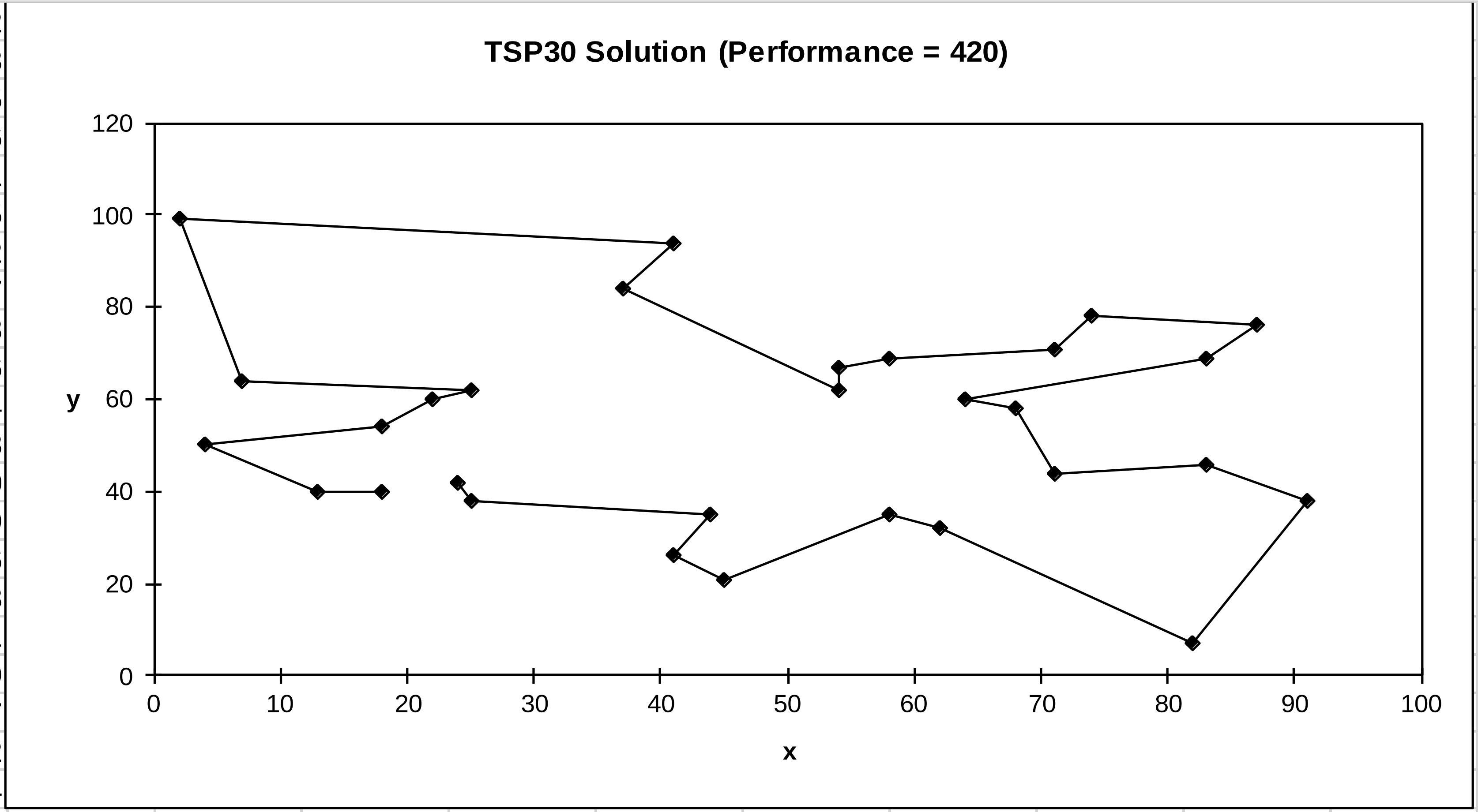


Solution $_k$(Distance = 652)

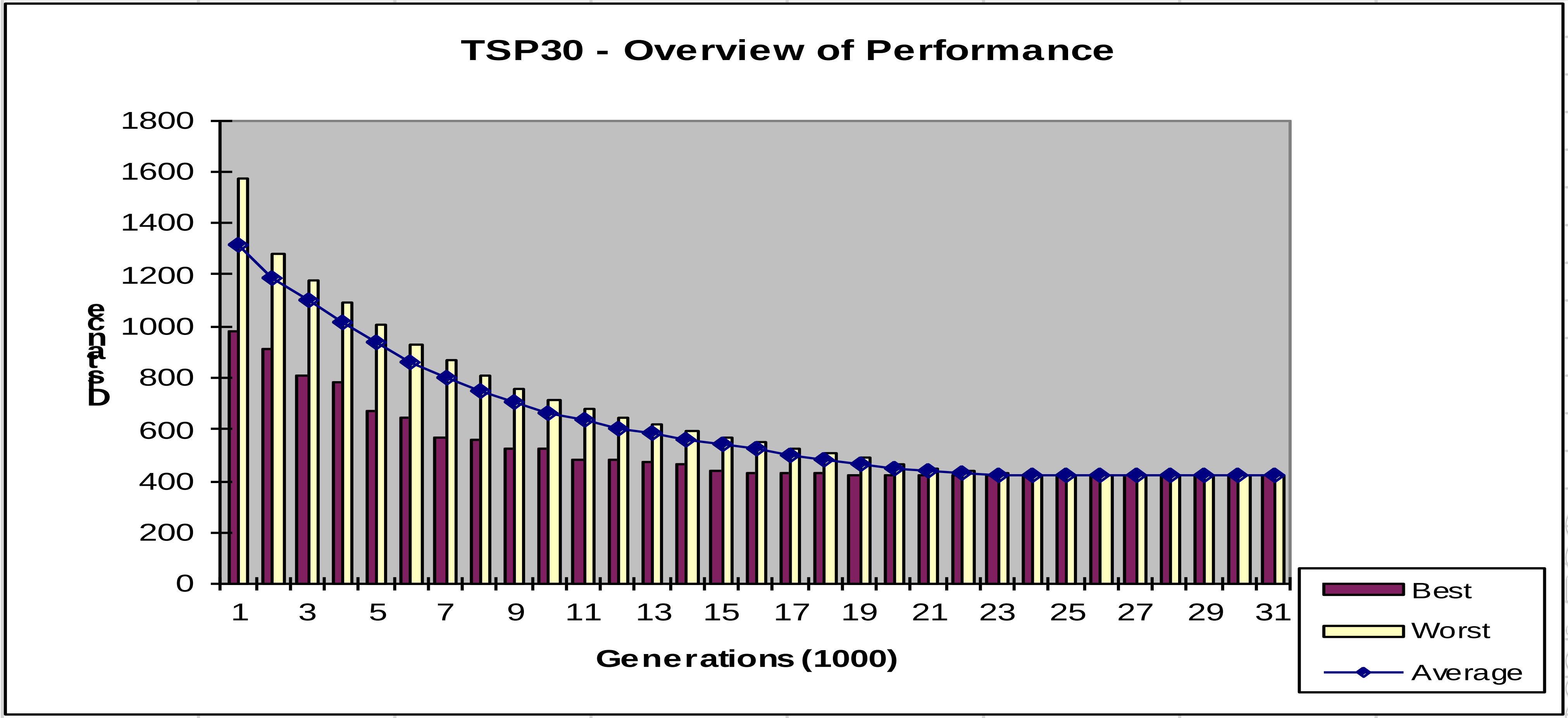


Best Solution (Distance = 420)

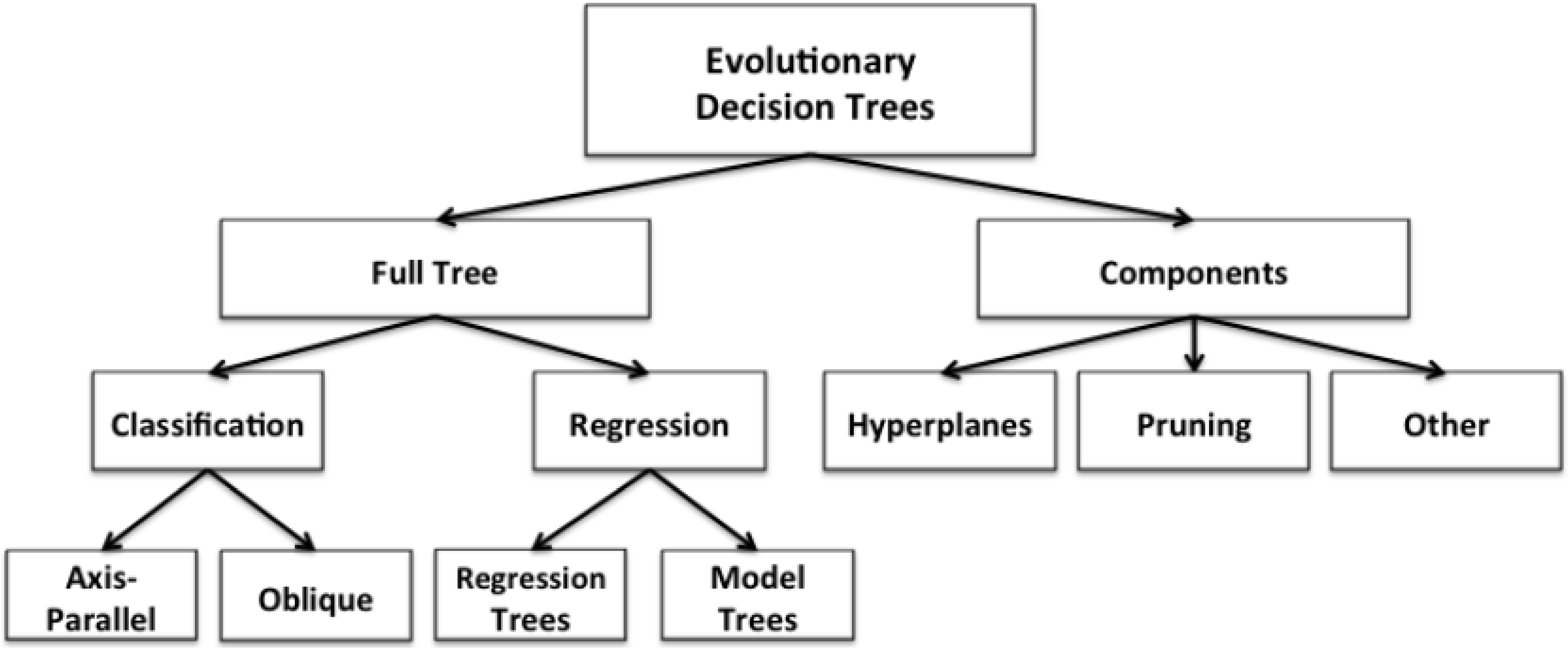
42
38
35
26
21
35
32
7
38
46
44
58
60
69
76
78
71
69
67
62
84
94



Overview of Performance



GA and Decision trees



- GA is slow but robust
- Gradient descent is fast but will not converge
- How do we have best of both

DEEP LEARNING

CSE 7303c

ANN died too

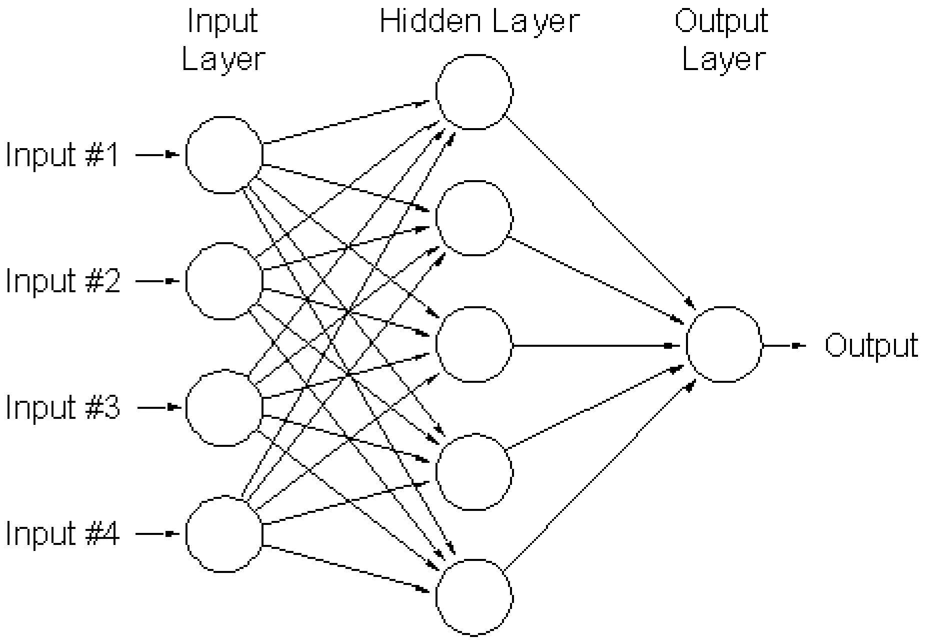
- http://www.newyorker.com/news/news-desk/is-deep-learning-a-revolution-in-artificial-intelligence

- They learned slowly and inefficiently, and as Steven Pinker and I showed, couldn’t master even some of the basic things that children do, like learning the past tense of regular verbs. By the late nineteen-nineties, neural networks had again begun to fall out of favor.

- They need trained samples and that is not how we learn

CSE 7303c

Brain and Artificial Neuralnet



Brain seems to be creating refined features slowly and systematically
More layers enable
In ANN with one or two layers, you cannot create such features

Adding more layers

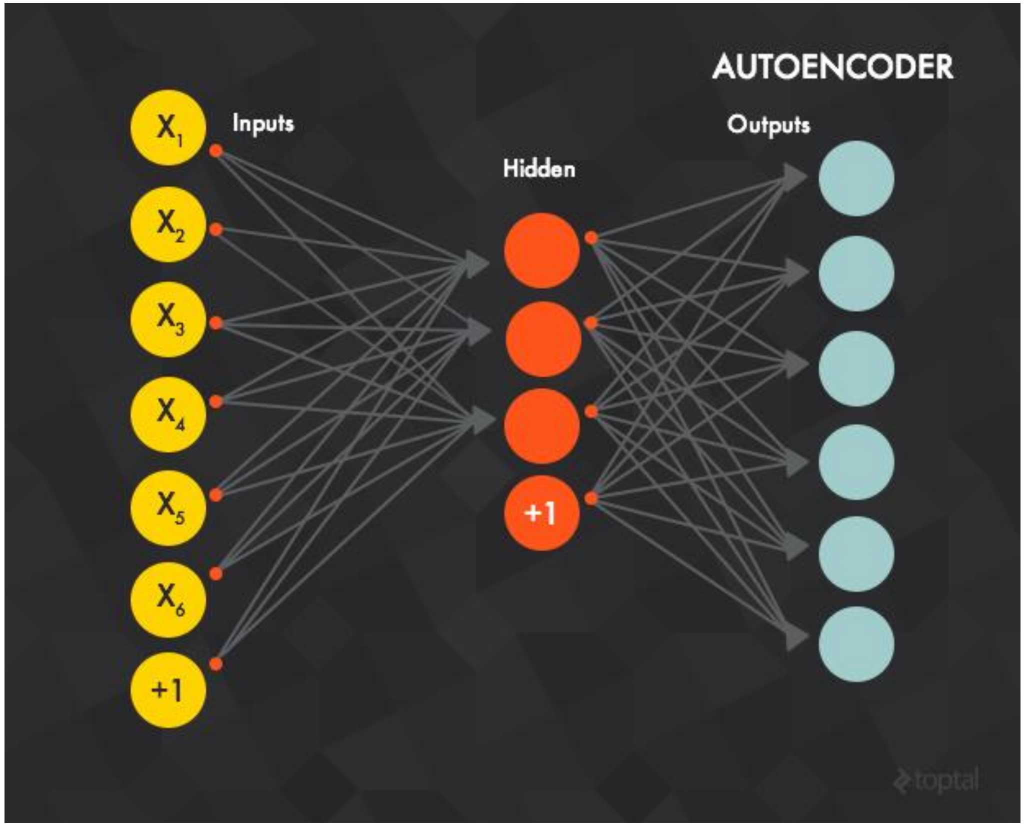
- Vanishing gradients: as we add more and more hidden layers, backpropagation becomes less and less useful in passing information to the lower layers. In effect, as information is passed back, the gradients begin to vanish and become small relative to the weights of the networks.

- Overfitting: perhaps the central problem in Machine Learning. Briefly, overfitting describes the phenomenon of fitting the training data *too* closely, maybe with hypotheses that are *too* complex. In such a case, your learner ends up fitting the training data really well, but will perform much, much more poorly on real examples.

How to add more layers

- Unsupervised learning to create features

Autoencoders



The network is trained to “recreate” the input. An autoencoder is a feed forward neural network which aims to *learn a compressed, distributed representation (encoding) of a dataset.*

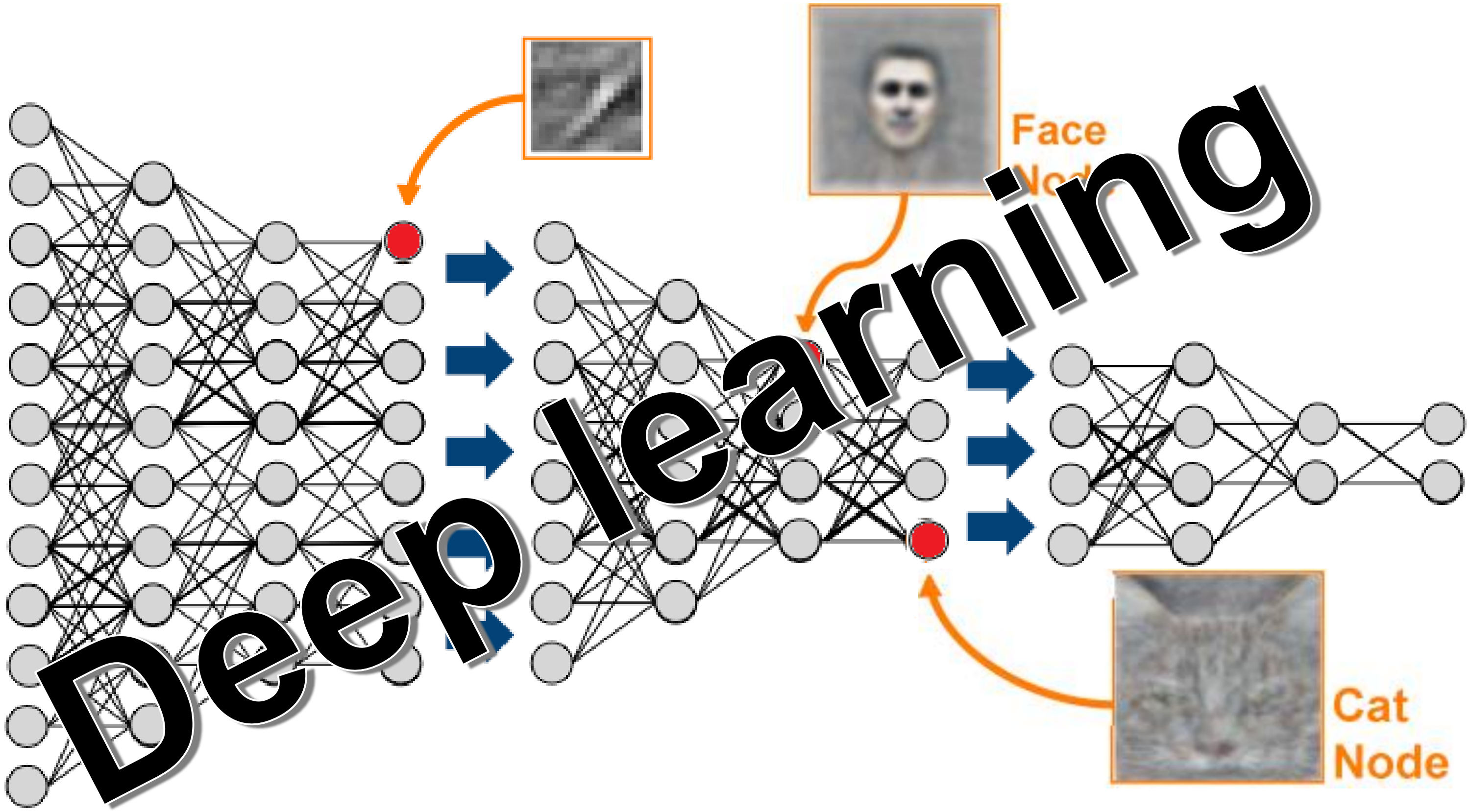
CSE 7303c

Autoencoders

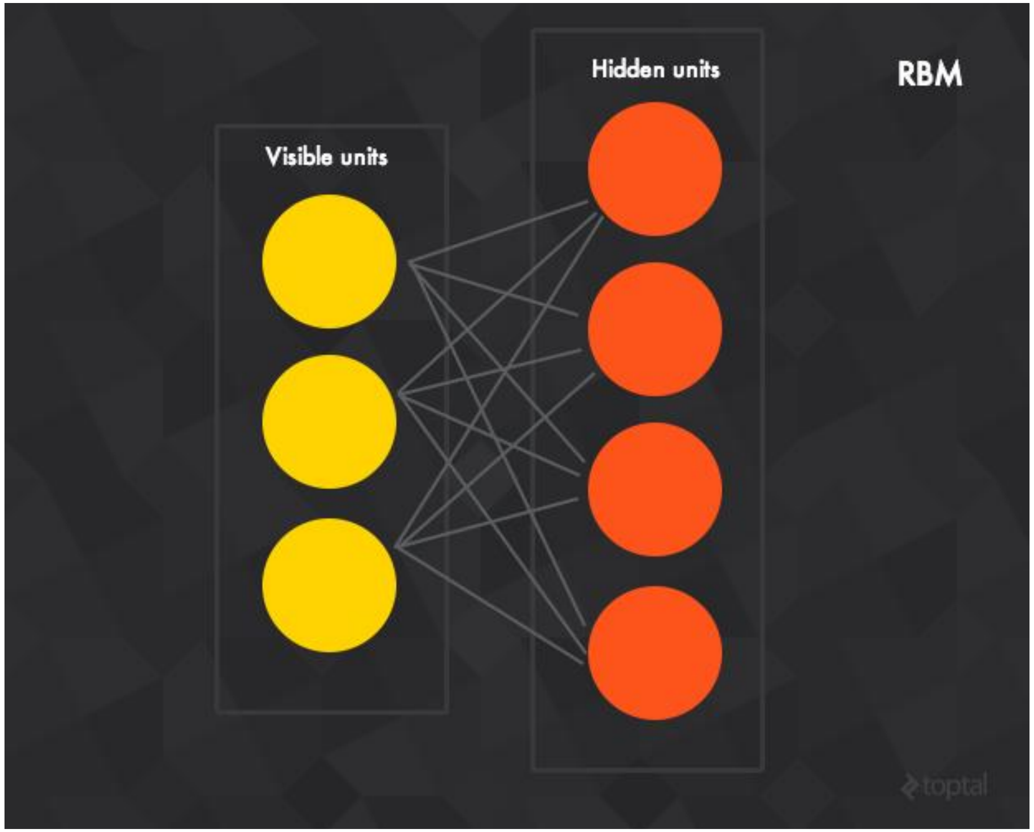
- The intuition behind this architecture is that the network will not learn a “mapping” between the training data and its labels, but will instead learn the *internal structure* and features of the data itself.

- Usually, the number of hidden units is smaller than the input/output layers, which forces the network to learn only the most important features and achieves a dimensionality reduction.

Deep learning



Restricted boltzman machines



In their simple form, they are binary

Activation energy $a_i = \sum w_{ij} x_j$
$p_i = \sigma(a_i)$, where σ is the is the logistic function.
Turn unit *i* on with probability of p_i

Output is stable to input changes

Learning in RBMs:

http://blog.echen.me/2011/07/18/introduction-to-restricted-boltzmann-machines/
http://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks

- Positive phase
  - Update the states of the hidden units using the logistic activation rule described above
  - Then for each edge e_{ij}, compute positive (e_{ij})=$x_i x_j$ (i.e., for each pair of units, measure whether they're both on).

Negative phase

- Reconstruct the visible units in a similar manner: for each visible unit, compute its activation energy a_i, and update its state. (Note that this *reconstruction* may not match the original preferences.)
- Then update the hidden units again, and compute Negative(e_{ij})=$x_i x_j$ for each edge.

CSE 7303c

Weight updates

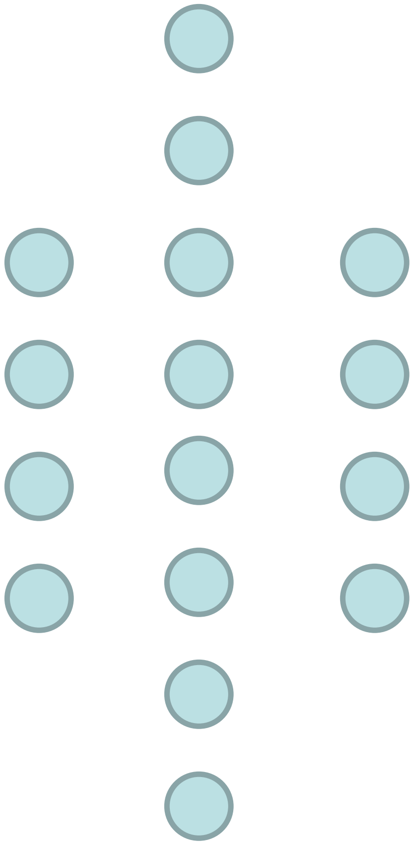
- Update the weight of each edge e_{ij} by setting $w_{ij}=w_{ij}+L(\text{Positive}(e_{ij})-\text{Negative}(e_{ij}))$, where L is a learning rate.
- Repeat over all training examples.
- Continue until the network converges (i.e., the error between the training examples and their reconstructions falls below some threshold) or we reach some maximum number of epochs.

Why does this make sense

- Positive(e_{ij}) measures the association between the ith and jth unit that we *want* the network to learn from our training examples;
- Negative(e_{ij}) measures the association that the network *itself* generates (or "daydreams" about) when no units are fixed to training data.
- So by adding Positive(e_{ij})−Negative(e_{ij}) to each edge weight, we're helping the network's daydreams better match the reality of our training examples.
- This update rule is called contrastive divergence, which is basically a funky term for "approximate gradient descent".

Sparse autoencoders

Different records may need different features



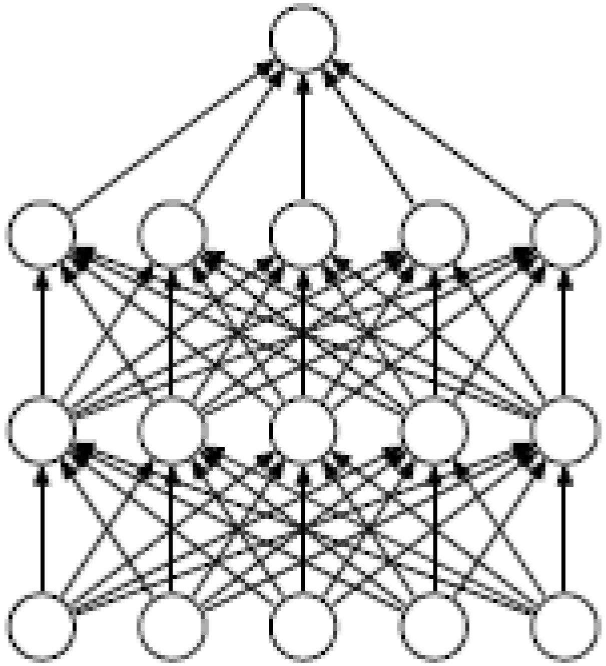
Sparse autoencoders

- What are the ideal properties of machine generated features
  - An example must be explained primarily by a few features
  - A feature must belong to only a few features
  - All features should have similar activities

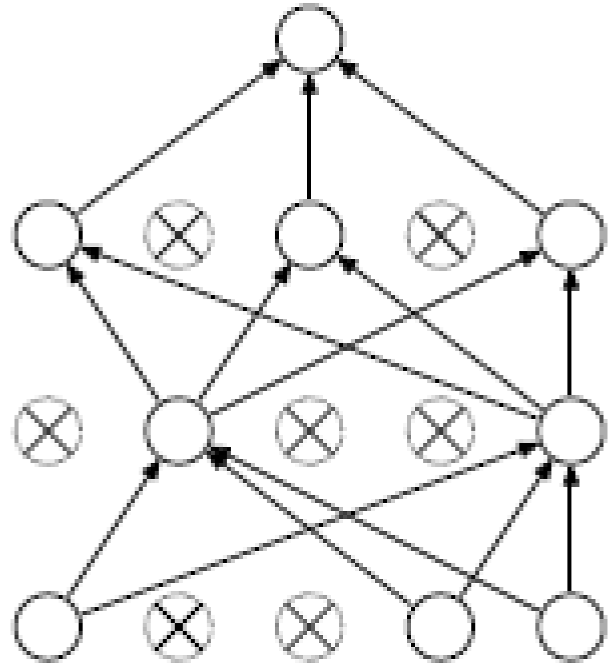
PRACTICAL TIPS

Drop outs

http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

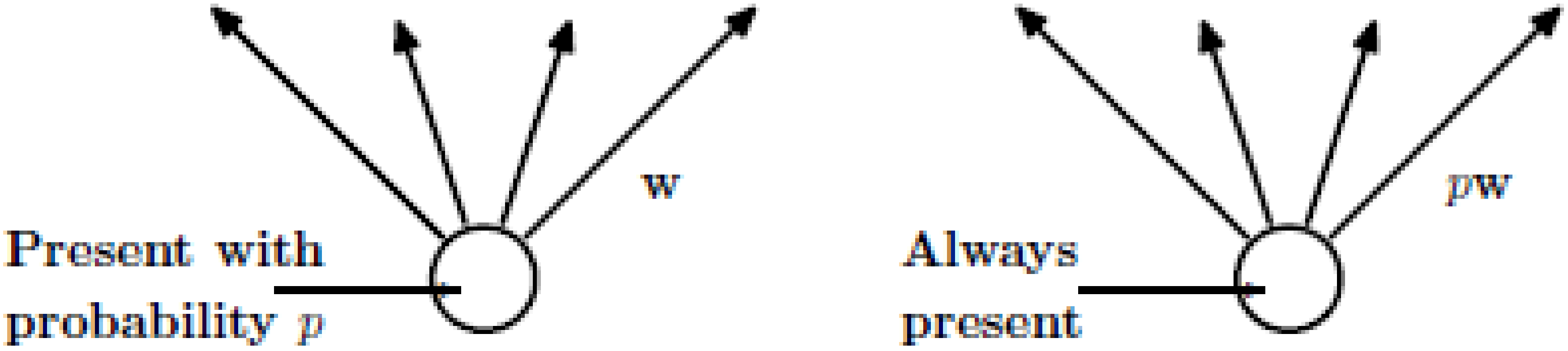


(a) Standard Neural Net



(b) After applying dropout.

Drop out



(a) At training time (b) At test time

Deep learners as predictive machines

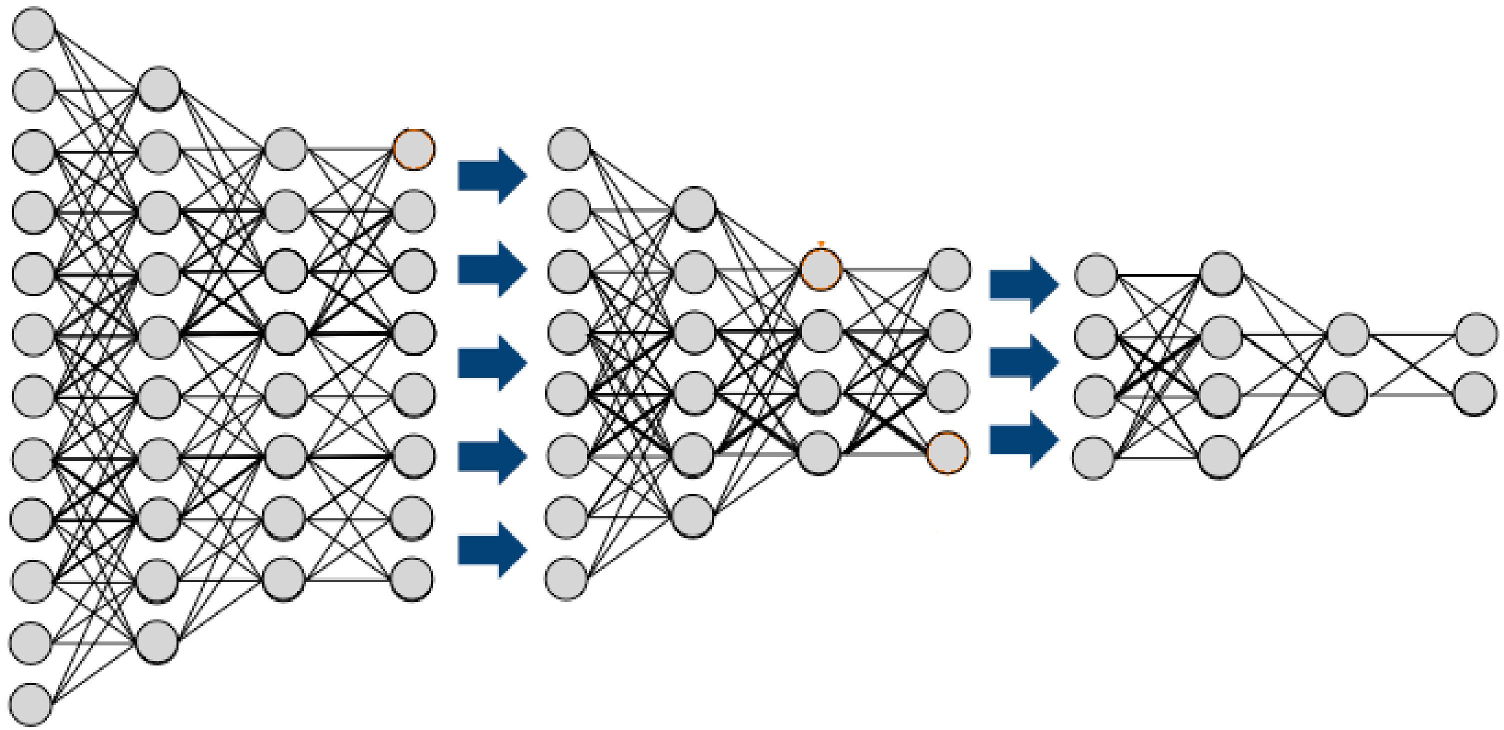
- Build a stacked autoencoders or RBMs

- Make the last layer as desired output

- Learn weights through one network level back propagation using already computed weights as initial weights

Stack them

- Hidden layer 1 becomes visible layer for the hidden layer 2 and one can keep stacking



CSE 7303c

Committing the same mistake

- http://ai.stanford.edu/~joni/papers/LasersonXRDS2011.pdf

CSE 7303c

A more interesting approach

- Use autoencoders, RBMs as feature generators

- Create 100s of features

- Run a simple linear model or a random forest (wopal wabbit or sofia)

Inspire...Educate...Transform.

International School of Engineering
2-56/2/19, Khanamet, Madhapur, Hyderabad - 500 081

For Individuals: +91-9177585755 or 040-65743991

For Corporates: +91-7893866005

Web: http://www.insofe.edu.in

Facebook: www.facebook.com/insofe/

LinkedIn: http://www.linkedin.com/groups/Big-Data-Analytics-Hadoop-Hyderabad-4488721?trk=myg_ugrp_ovr

YouTube: http://www.youtube.com/InsofeVideos

SlideShare: http://www.slideshare.net/INSOFE