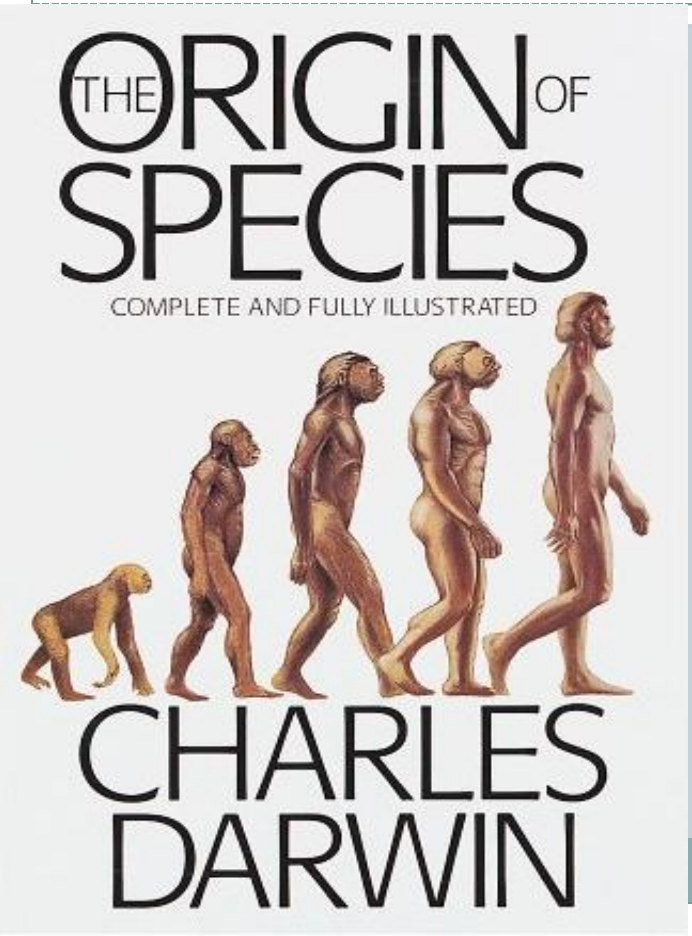
Genetic Algorithms

Objectives

To provide a background and understanding of basic genetic algorithms and some of their applications.

- a basic genetic algorithm
- the basic discussion
- the applications of the algorithm

Genetic Algorithms



Charles Darwin

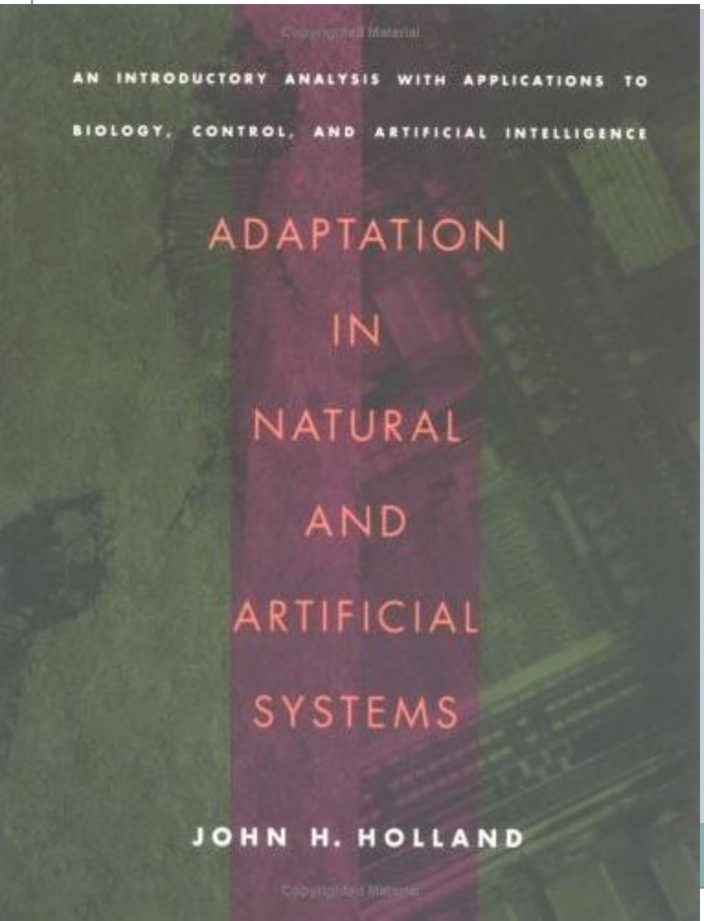
1859

Origin of the Species

Survival of the Fittest

Genetic Algorithms

John Holland



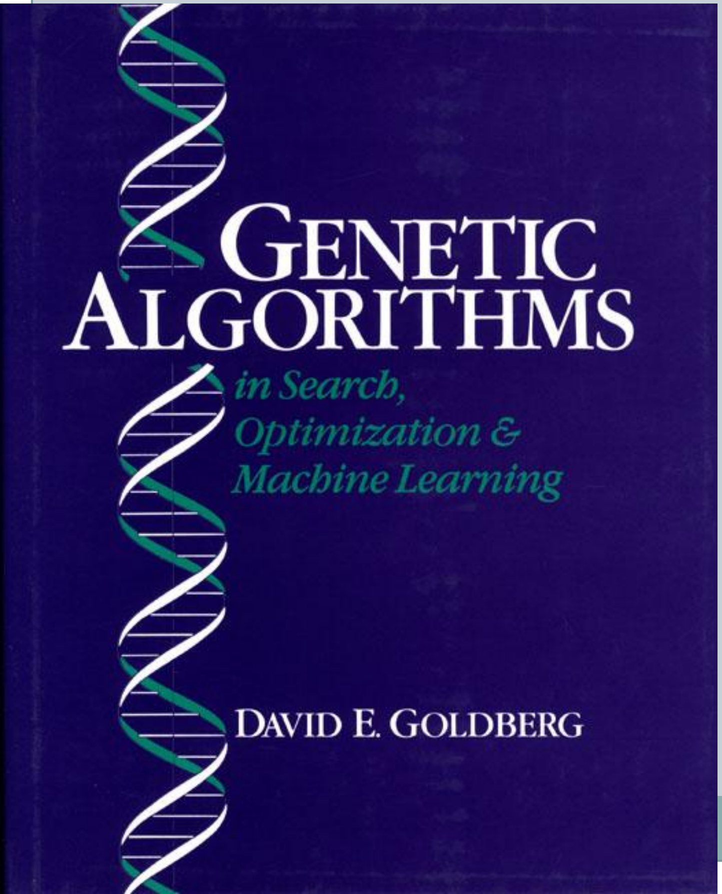
1975

Genetic Algorithms

Artificial Survival of the Fittest

Genetic Algorithms

David Goldberg



1989

Genetic Algorithms

Foundations and Applications

Genetic Algorithms

Terms

Biological	GA Terms
Chromosome	String
Gene	Feature, character
Genomes	Guesses, solutions, collection of genes

Genetic algorithms are search procedures based on the mechanics of genetics and natural selection.

Genetic Algorithms

Compared to exhaustive tree search techniques, modern heuristic algorithms quickly converge to sub-optimal solutions after examining only a small fraction of the **search space** and have been successfully applied to complex optimisation problems.

Genetic Algorithms

Genetic algorithms try to imitate the Darwinian evolution process in computer programs.

In evolutionary systems, **populations** evolve by selective pressures, mating between individuals, and alterations such as mutations.

In genetic algorithms, **genetic operators** evolve solutions in the current population to create a new population, simulating similar effects.

Genetic Algorithms

A genetic algorithm simulates Darwinian theory of evolution using highly parallel, mathematical algorithms that, transform a set (**population**) of solutions (typically strings of 1's and 0's) into a new population, using operators such as: **reproduction**, **mutation** and **crossover**.

Genetic Algorithms

- Determine the initial population of creatures
- Determine the fitness of the population
- Reproduce the population using the fittest parents of the last generation
- Determine the crossover point, this can also be random
- Determine if mutation occurs and if so on which creature(s)
- Repeat from step 2 with the new population until condition (X) is true

Genetic Algorithms

- Can be implemented as three modules; the **evaluation** module, the **population** module and the **reproduction** module
  - Initial population
  - Evaluations on individuals
  - Reproduction
    - Choose suitable parents (by evaluation rating)
    - Produce two offspring (by probability)
    - Mutation
  - *Domain knowledge – evaluation function*

Genetic Algorithms
Encoding the Solutions

The decision variables of a problem are normally encoded into a finite length string

This could be a binary string or a list of integers

For example :

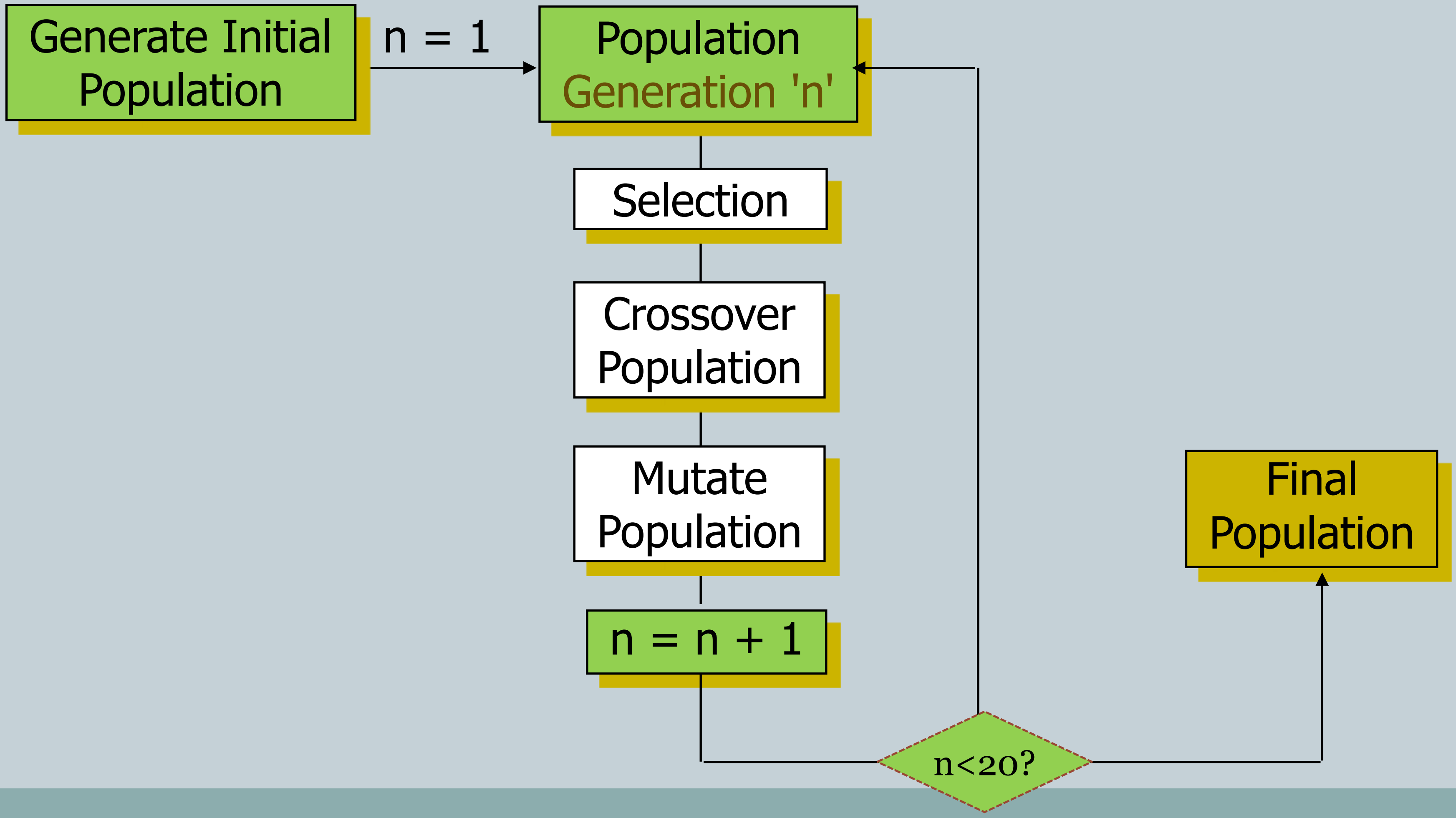
| 0 1 1 0 1 1 0 1 0 |

or

5	1	4	2	3	4	1

We could also represent numbers as coloured boxes

Genetic Algorithms Flowchart (20 generations)

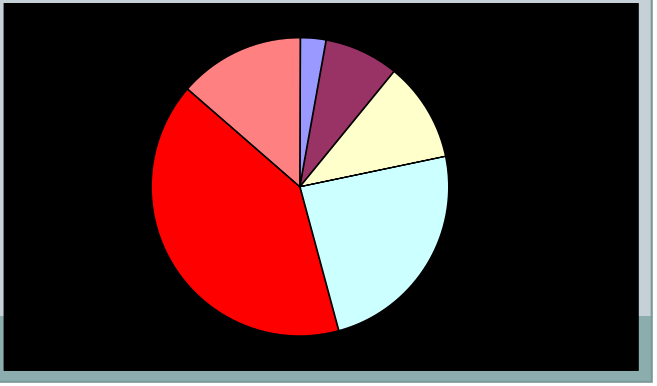


Genetic Algorithms

Parent selection

- Sum the fitness of all the population members, *TF*
- Generate a random number, *m*, between 0 and *TF*
- Return the first population member whose fitness added to the preceding population members is greater than or equal to *m*

Roulette Wheel Selection

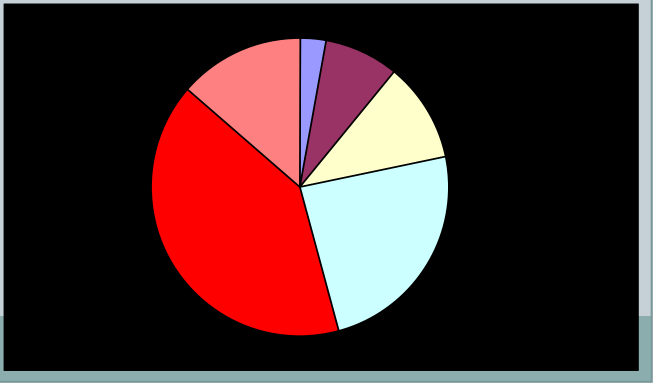


Genetic Algorithms

Parent selection

Evaluation

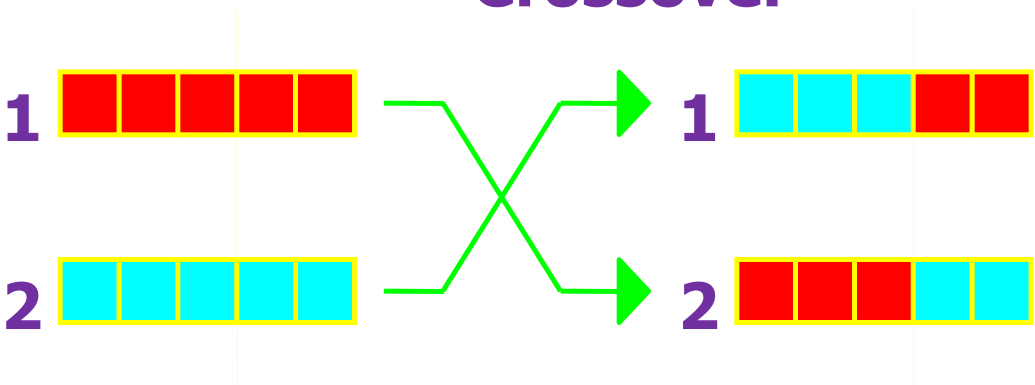
- Responsible for evaluating a chromosome
- Only part of the GA that has any knowledge about the problem. The rest of the GA modules are simply operating on (typically) bit strings with no information about the problem
- A different evaluation module is needed for each problem



Genetic Algorithms

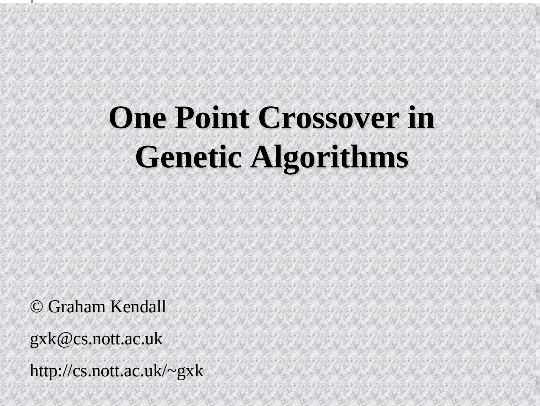
Crossover

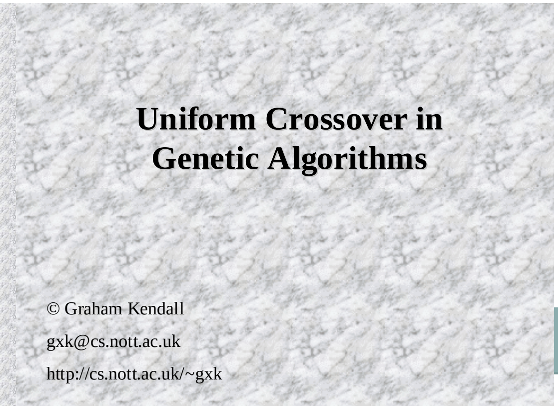
Crossover



A percentage of the population is selected for breeding and assigned random mates.

A random crossover point is selected and a pair of new solutions are produced

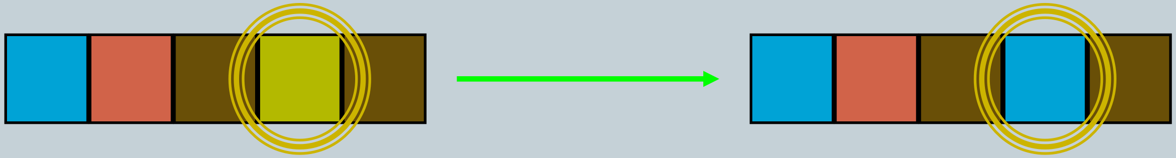




Adapted from Dr Graham Kendall's G5BAIM lecture example

Genetic Algorithms

Mutation



In a small percentage of the population random changes are made.

Genetic Algorithms

Each iteration of the loop is called a **generation**, **fitness** can be gauged in several different ways depending on the application of the algorithm.

Reproduction and crossover play the primary role in artificial genetic search. Reproduction emphasises highly fit strings (or good solutions), and crossover recombines these selected solutions for new, potentially better solutions.

Mutation plays a secondary role in the search by providing diversity in the population.

Genetic Algorithms

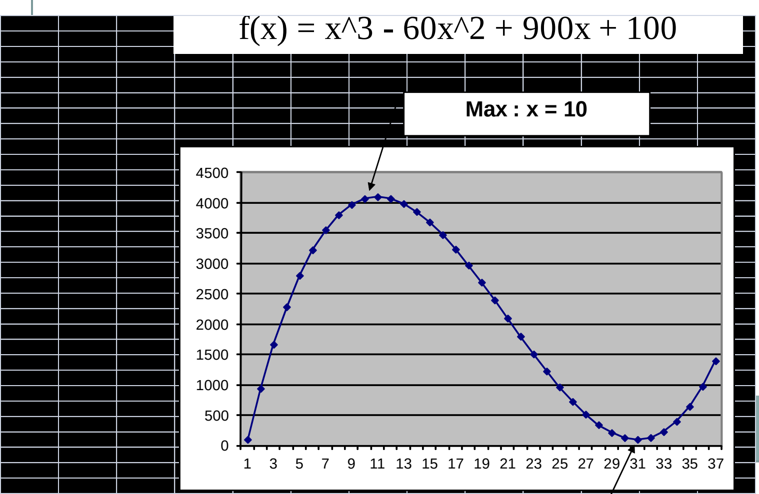
Conceivable solutions are represented, the **'fittest'** will have the greatest chance of reproducing.

Successful properties will therefore be favourably represented within the **population** of solutions. The population is the successively 'optimised' after a number of generations.

Genetic Algorithm Example I

Maximise f(x) = x3 - 60 * x2 + 900 * x +100
0 <= x <= 31

Crossover probability, PC = 1.0
Mutation probability, PM = 0.0
x can be represented using five binary digits



Genetic Algorithm Example I

Generating random initial population

Maximise f(x) = x3 - 60 * x2 + 900 * x +100 (0 <= x <= 31)

chromosome	**binary string**	**x**	**f(x)**
P1	11100	28	212
P2	01111	15	3475
P3	10111	23	1227
P4	00100	4	2804
	Total		7718
	Average		1929.50

Genetic Algorithm Example I

Choose two parents (roulette selection)

Roulette wheel	Parents
4116	P3
1915	P2

Genetic Algorithm Example I

One point crossover (random position 1)

P_3	1	0	1	1	1
P_2	0	1	1	1	1

C_1	1	1	1	1	1
C_2	0	0	1	1	1

P_4	0	0	1	0	0
P_2	0	1	1	1	1

C_3	0	0	1	1	1
C_4	0	1	1	0	0

Genetic Algorithm Example I

New generation

Maximise f(x) = x3 - 60 * x2 + 900 * x +100 (0 <= x <= 31)

chromosome	binary string	x	f(x)
P1	11111	31	131
P2	00111	7	3803
P3	00111	7	3803
P4	01100	12	3889
	Total		11735
	Average		2933.75

Genetic Algorithm Example I

Two generations

Maximise f(x) = x3 - 60 * x2 + 900 * x +100 (0 <= x <= 31)

chromosome	binary string	x	f(x)
P1	11100	28	212
P2	01111	15	3475
P3	10111	23	1227
P4	00100	4	2804
	Total		7718
	Average		**1929.50**

chromosome	binary string	x	f(x)
P1	11111	31	131
P2	00111	7	3803
P3	00111	7	3803
P4	01100	12	3889
	Total		11735
	Average		**2933.75**

Mutation

Genetic Algorithm Example II

Traveling Salesman Problem

a number of cities

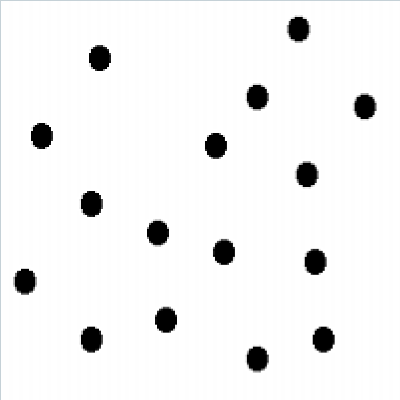
costs of traveling between cities

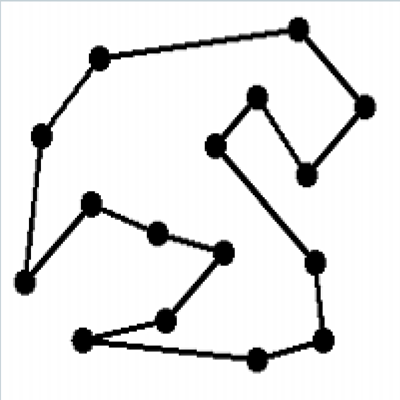
a traveling sales man needs to visit all these cities exactly once and return to the starting city

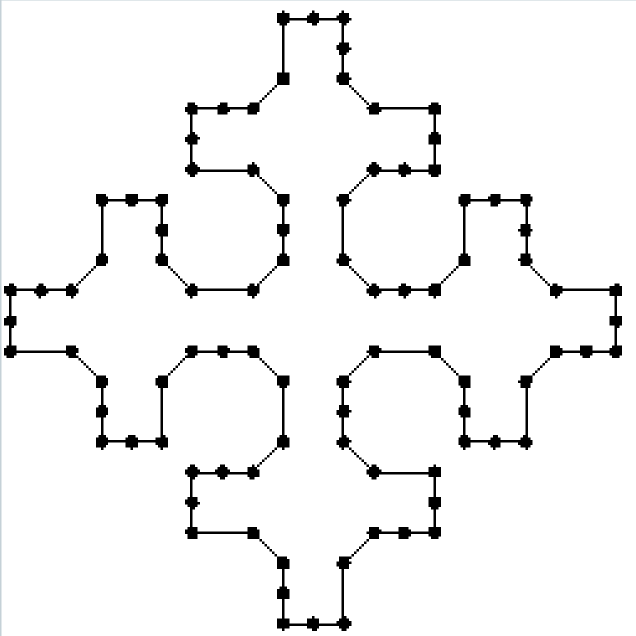
What’s the cheapest route?

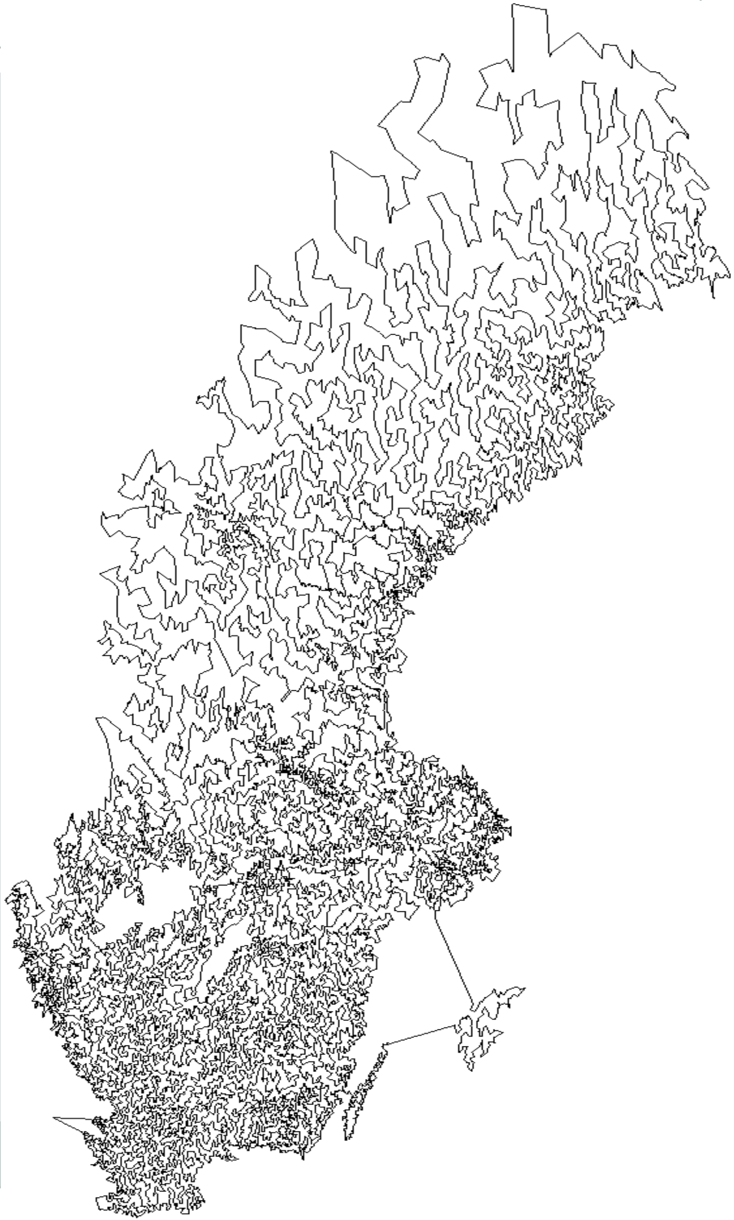
Genetic Algorithm Example II

Traveling Salesman Problem





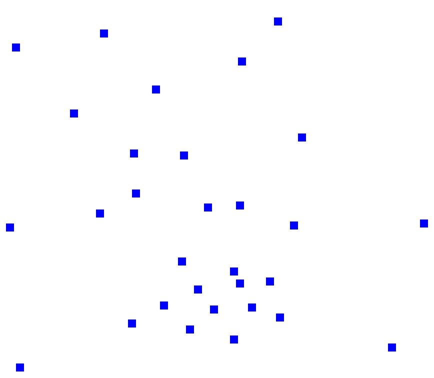




Genetic Algorithm Example II

Initial generation

P_1	5	8	1	…	…	84	32	27	54	67	6.5
P_2	78	81	27	…	…	9	11	7	44	24	7.8
			…								
P_{30}	8	1	7	…	…	9	16	36	24	19	6.0



Genetic Algorithm Example II

Choose pairs of parents

P_{30}	8	1	7	…	…	9	16	36	24	19	6.0
P_2	78	81	27	…	…	9	11	7	44	24	7.8

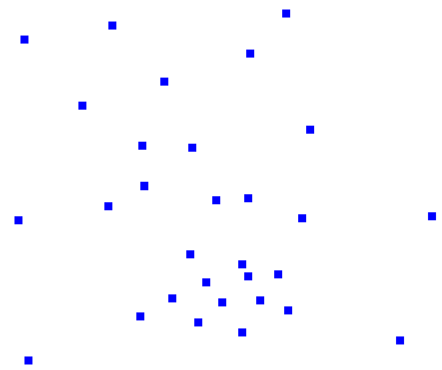
Crossover

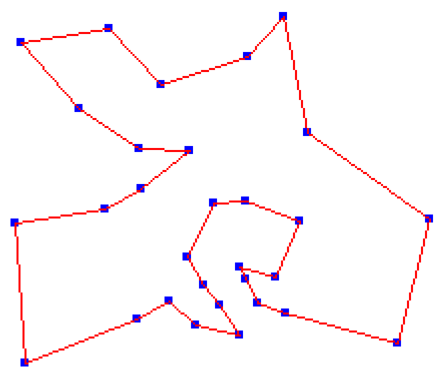
C_1	8	1	7	…	…	9	11	13	44	24	5.9
C_2	78	81	27	…	…	9	16	36	24	19	6.2

Genetic Algorithm Example II

Next generation

P_1	8	1	7	…	…	9	11	7	44	24	5.9
P_2	78	81	27	…	…	9	16	36	24	19	6.2
			...								
P_2	7	8	2	…	…	5	10	76	4	79	6.0



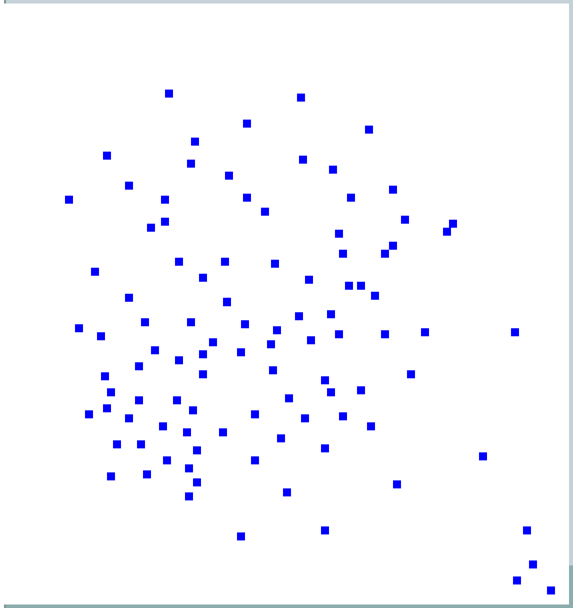


Genetic Algorithm Example II

Traveling Salesman Problem

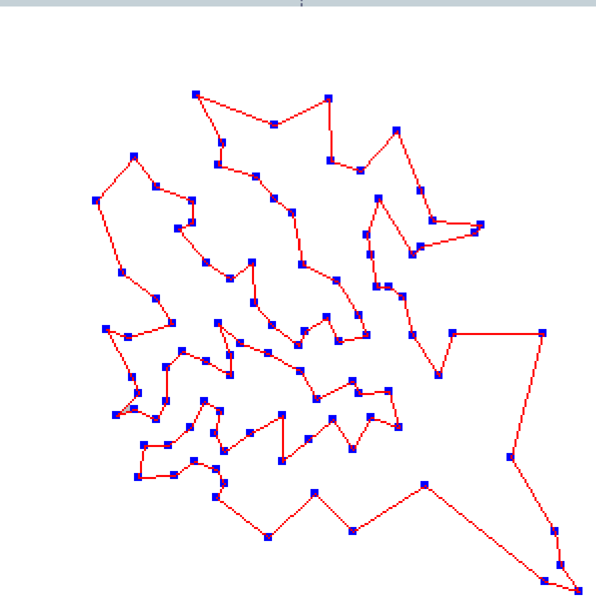
No. of cities: 100

Population size: 30



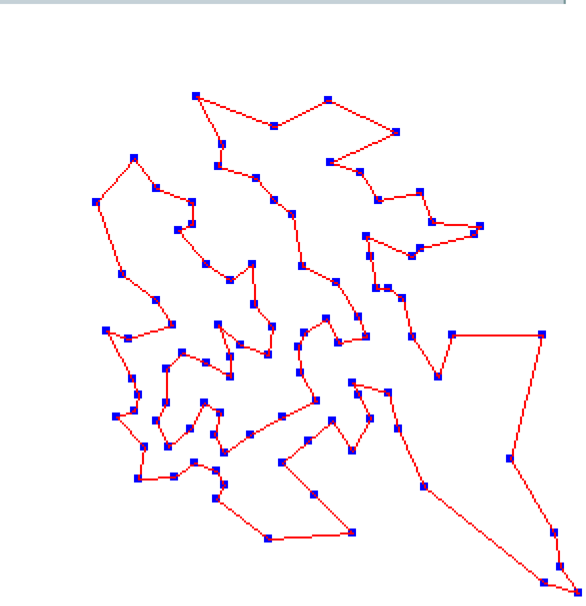
Cost: 6.37

Generation: 88



Cost: 6.34

Generation: 1100



Genetic Algorithm Example III

There are many diverse applications of genetic algorithms. They are best suited to problems where the efficient solutions are not already known.

If they are applied to solvable problems, they will be easily out-performed by efficient standard computing methods.

The strength of GA's is their ability to heuristically search for solutions when all else fails. If you can **represent** the solutions to the problem in a suitable format, such as a series of 1's and 0's, then the GA will do the rest.

Genetic Algorithm Example III

Applying Genetic Algorithms to Personnel Scheduling

Personnel scheduling in healthcare is usually a very complex operation which has a profound effect upon the efficient usage of expensive resources.

Genetic Algorithm Example III

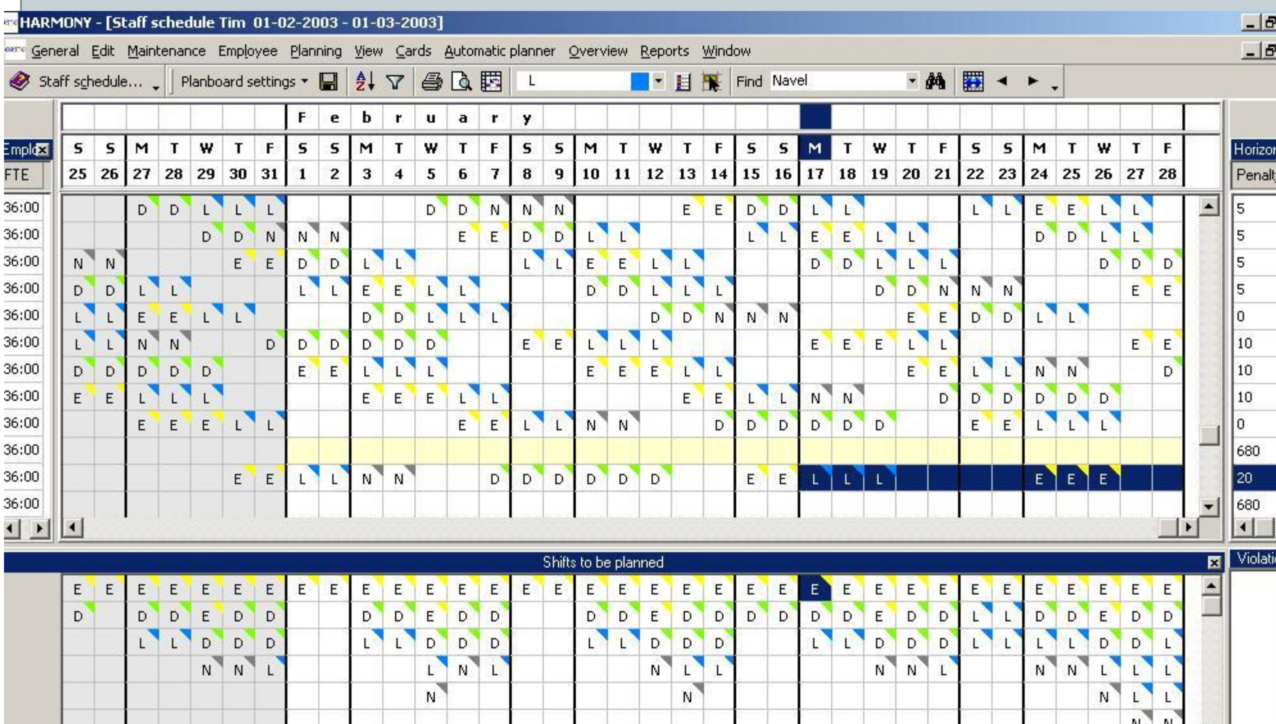
A number of nurses

A number of shifts each day

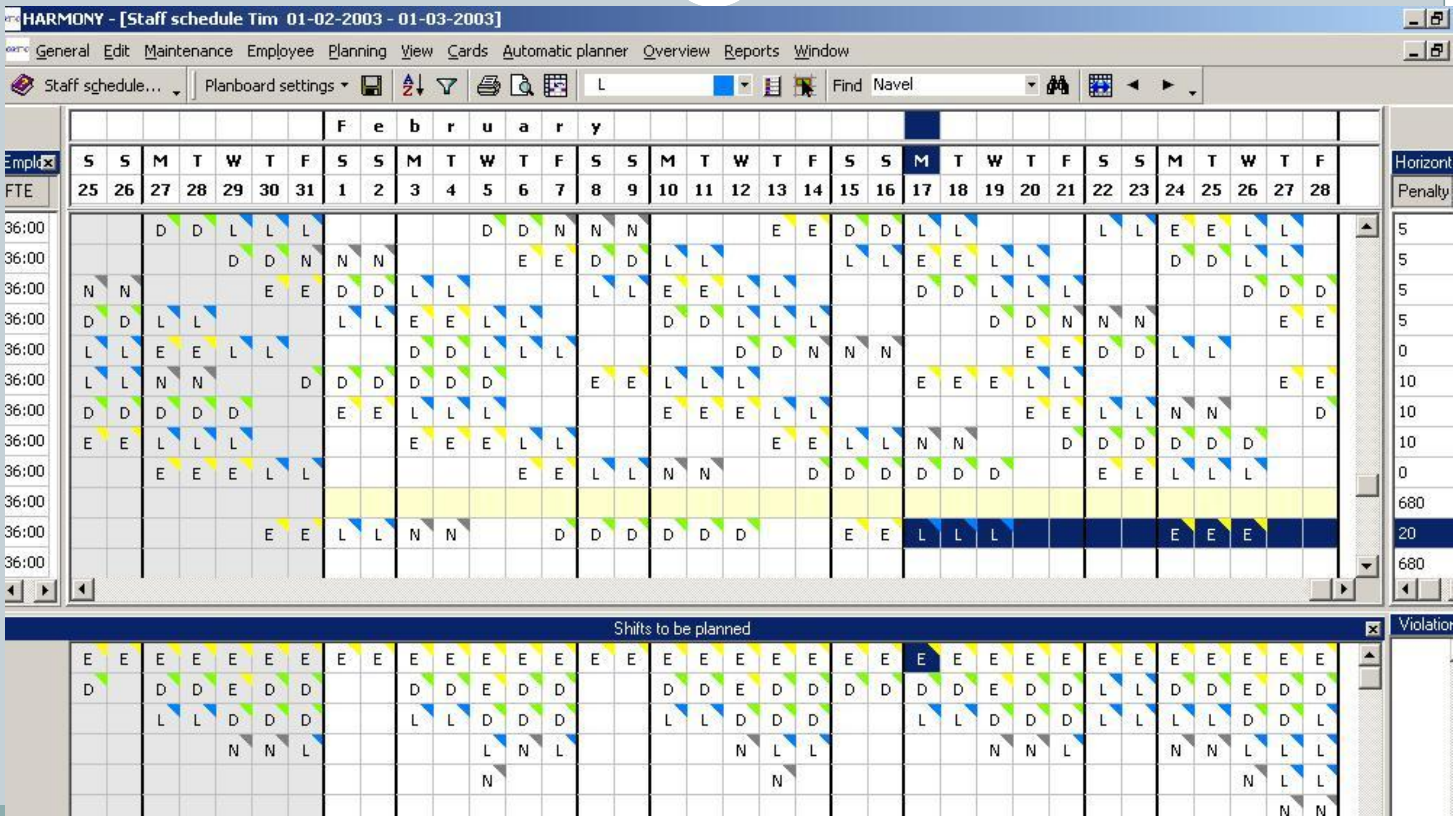
A set of constraints

- shift coverage
- one shift per day

- resting time
- workload per month
- consecutive shifts
- working weekends
- …



Genetic Algorithm Example III



Genetic Algorithm Example III

Genetic Algorithm

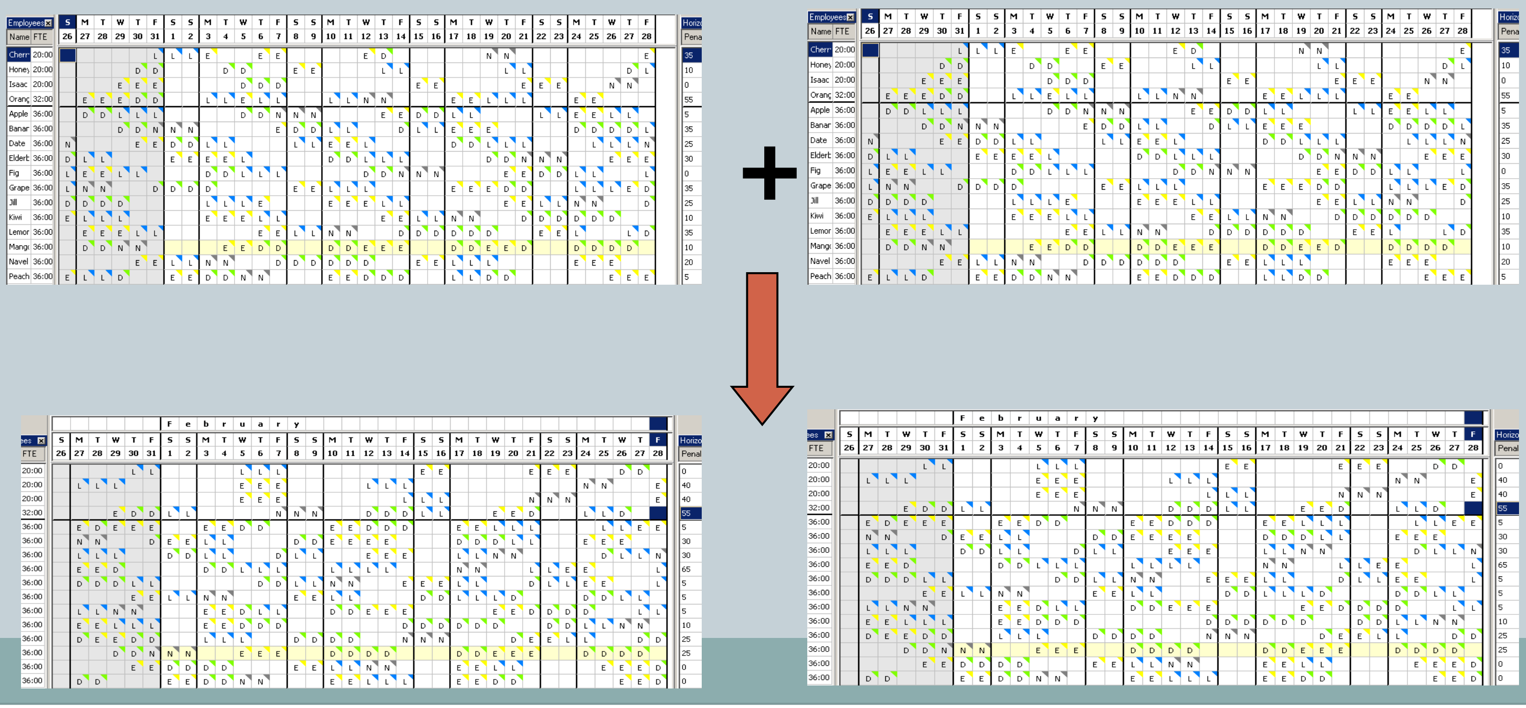
- -Initial population
  - -construct rosters
  - -repair infeasible ones

Employees		S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	Horiz
Name	FTE	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Pena
Cherr	20:00						L	L	L	E			E	E					E	D						N	N								E	35
Honey	20:00					D	D				D	D			E	E				L	L						L	L						D	L	10
Isaac	20:00				E	E	E					D	D	D								E	E					E	E	E			N	N		0
Orang	32:00		E	E	E	D	D			L	L	E	L	L			L	L	N	N				E	E	L	L	L			E	E				55
Apple	36:00		D	D	L	L	L					D	D	N	N	N				E	E	D	D	L	L				L	L	E	E	L	L		5
Banar	36:00				D	D	N	N	N					E	D	D	L	L			D	L	L	E	E	E					D	D	D	D	L	35
Date	36:00	N				E	E	D	D	L	L				L	L	E	E	L					D	D	L	L	L				L	L	L	N	25
Eldert	36:00	D	L	L				E	E	E	E	L					D	D	L	L	L					D	D	N	N	N			E	E	E	30
Fig	36:00	L	E	E	L	L				D	D	L	L	L					D	D	N	N	N				E	E	D	D	L	L			L	0
Grape	36:00	L	N	N			D	D	D	D					E	E	L	L	L					E	E	E	D	D			L	L	L	E	D	35
Jill	36:00	D	D	D	D					L	L	L	E				E	E	E	L	L						E	E	L	L	N	N			D	25
Kiwi	36:00	E	L	L	L					E	E	E	L	L						E	E	L	L	N	N			D	D	D	D	D	D			10
Lemor	36:00		E	E	E	L	L						E	E	L	L	N	N			D	D	D	D	D	D			E	E	L			L	D	35
Mang	36:00		D	D	N	N					E	E	D	D			D	D	E	E	E			D	D	E	E	D			D	D	D	D		10
Navel	36:00					E	E	L	L	N	N			D	D	D	D	D	D			E	E	L	L	L					E	E	E			20
Peach	36:00	E	L	L	D			E	E	D	D	N	N				E	E	D	D	D			L	L	D	D						E	E	E	5

Genetic Algorithm Example III

Genetic Algorithm

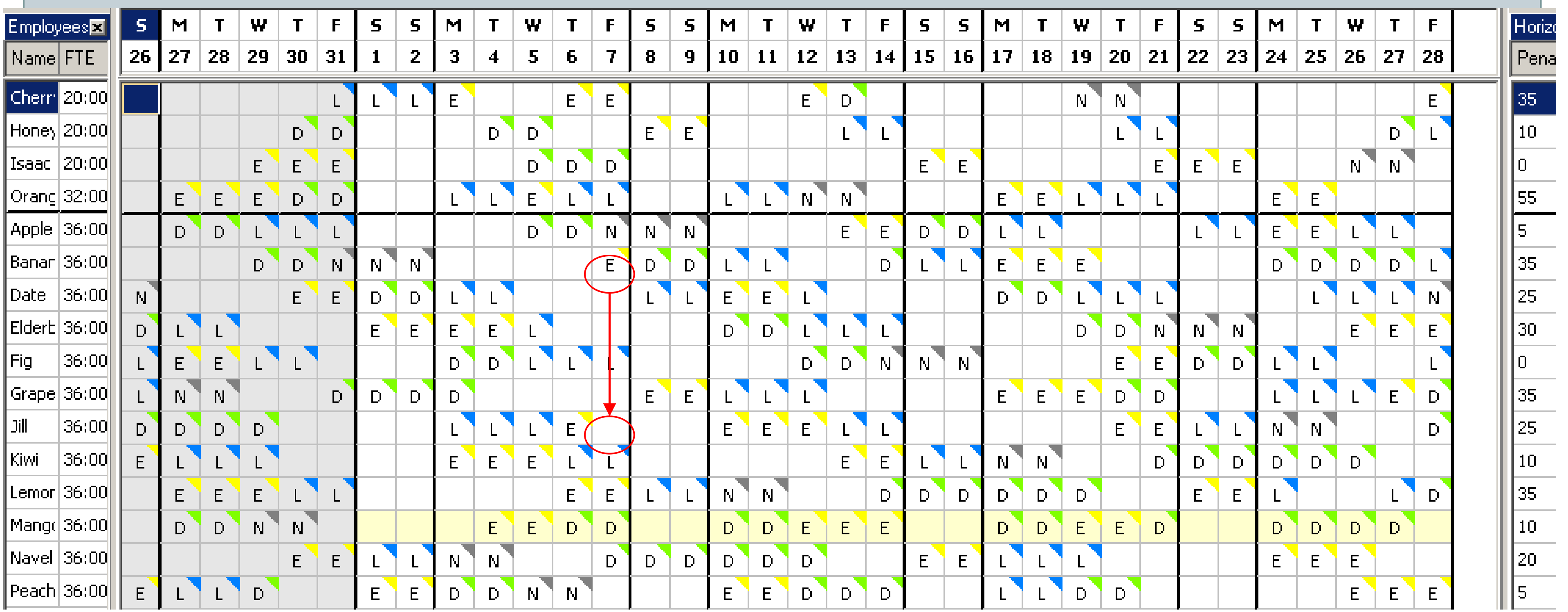
- -Select parents
- -Recombine rows in the two rosters
  - -repair infeasible ones



Genetic Algorithm Example III

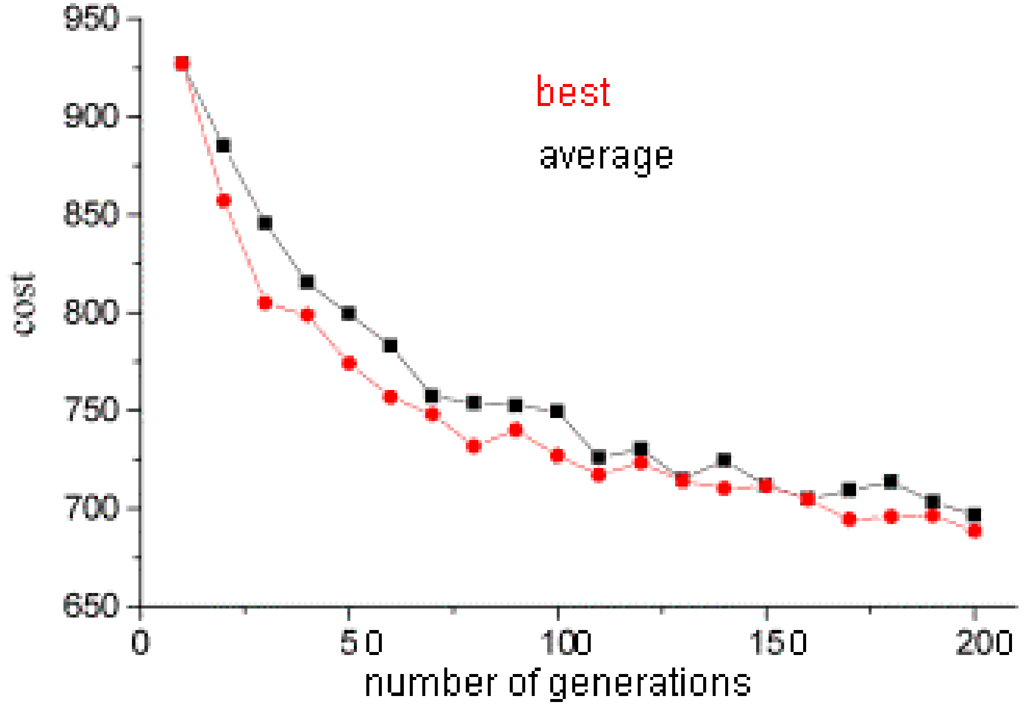
Genetic Algorithm

- -Mutation
- -Local optimiser



Name	FTE	S 26	M 27	T 28	W 29	T 30	F 31	S 1	S 2	M 3	T 4	W 5	T 6	F 7	S 8	S 9	M 10	T 11	W 12	T 13	F 14	S 15	S 16	M 17	T 18	W 19	T 20	F 21	S 22	S 23	M 24	T 25	W 26	T 27	F 28	Pena
Cherr	20:00						L	L	L	E			E	E					E	D						N	N								E	35
Hone	20:00					D	D				D	D			E	E				L	L						L	L						D	L	10
Isaac	20:00				E	E	E					D	D	D								E	E					E	E	E			N	N		0
Orang	32:00		E	E	E	D	D			L	L	E	L	L			L	L	N	N				E	E	L	L	L			E	E				55
Apple	36:00		D	D	L	L	L					D	D	N	N	N				E	E	D	D	L	L				L	L	E	E	L	L		5
Banar	36:00				D	D	N	N	N					E	D	D	L	L			D	L	L	E	E	E					D	D	D	D	L	35
Date	36:00	N				E	E	D	D	L	L				L	L	E	E	L					D	D	L	L	L				L	L	L	N	25
Eldert	36:00	D	L	L				E	E	E	E	L					D	D	L	L	L					D	D	N	N	N			E	E	E	30
Fig	36:00	L	E	E	L	L				D	D	L	L	L					D	D	N	N	N				E	E	D	D	L	L			L	0
Grape	36:00	L	N	N			D	D	D	D					E	E	L	L	L					E	E	E	D	D			L	L	L	E	D	35
Jill	36:00	D	D	D	D					L	L	L	E				E	E	E	L	L						E	E	L	L	N	N			D	25
Kiwi	36:00	E	L	L	L					E	E	E	L	L						E	E	L	L	N	N			D	D	D	D	D	D			10
Lemor	36:00		E	E	E	L	L						E	E	L	L	N	N			D	D	D	D	D	D			E	E	L			L	D	35
Mang	36:00		D	D	N	N					E	E	D	D			D	D	E	E	E			D	D	E	E	D			D	D	D	D		10
Navel	36:00					E	E	L	L	N	N			D	D	D	D	D	D			E	E	L	L	L					E	E	E			20
Peach	36:00	E	L	L	D			E	E	D	D	N	N				E	E	D	D	D			L	L	D	D						E	E	E	5

Genetic Algorithm Example III

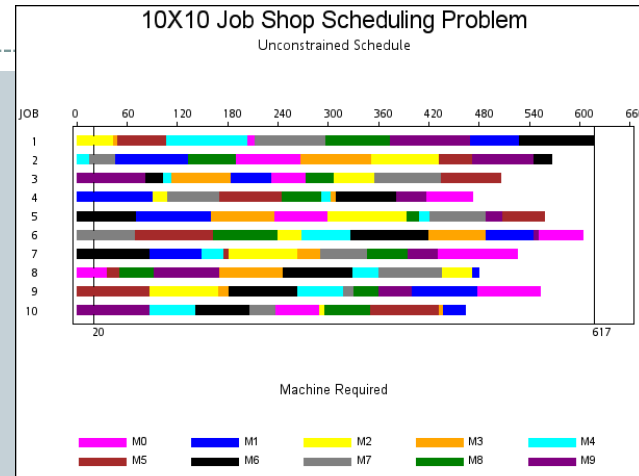


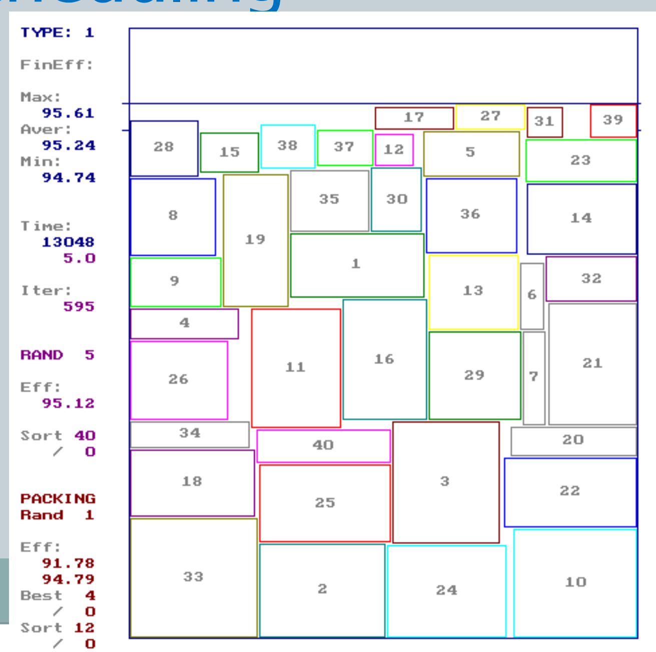
Population Size	50
Crossover Probability	0.7
Mutation Probability	0.001
Local Optimiser	ON

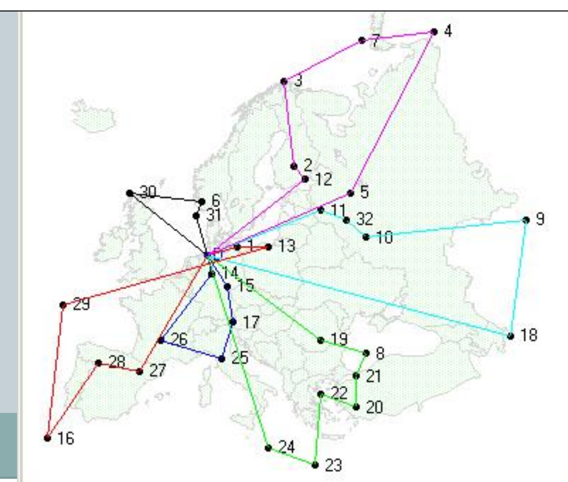
Genetic Algorithm Applications

Combinatorial optimisation problems

- bin packing problems
- vehicle routing problems
- job shop scheduling



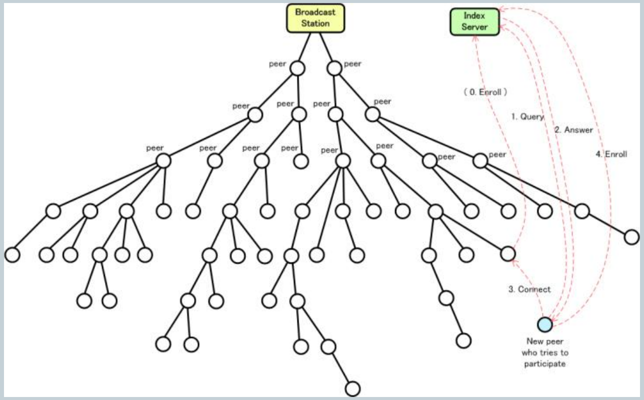


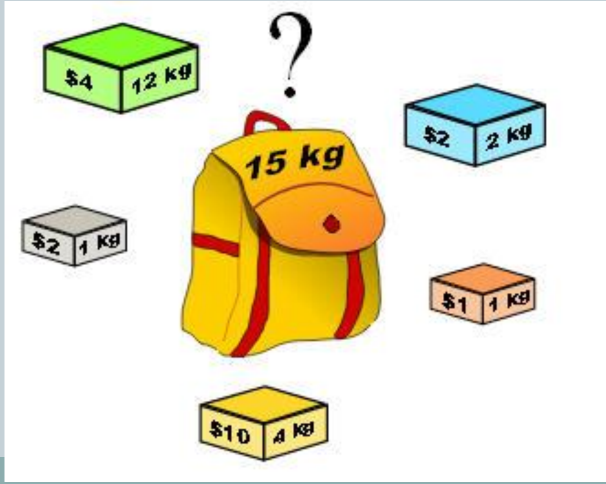


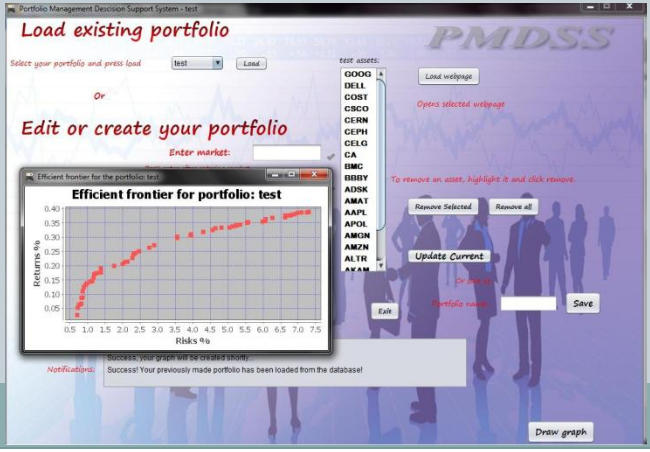
Genetic Algorithm Applications

Combinatorial optimisation problems

- portfolio optimization
- multimedia multicast routing
- knapsack problem







Genetic Algorithm Performance

There are a number of factors which affect the performance of a genetic algorithm.

- The size of the population
- Selection pressure (elitism, tournament)
- The cross-over probability
- The mutation probability
- Defining convergence
- Local optimisation

Genetic Algorithm in Meta-heuristics

Meta-heuristics

- Population based
  - Genetic algorithms
  - Memetic algorithms
  - Swarm intelligence (ant colony, particle swarm optimisation)
  - …
- Local search based
  - Tabu search
  - Variable neighbourhood search
  - Simulated annealing
  - …

Conclusions

Survival of the fittest, the most fit of a particular generation (the nearest to a correct answer) are used as parents for the next generation.

The most fit of this new generation are parents for the next, and so on. This eradicates worse solutions and eliminates bad family lines.

Conclusions

Genetic algorithms are a very powerful tool, allowing searching for solutions when there are no other feasible means to do so.

The algorithm is easy to produce and simple to understand and enhancements easily introduced. This gives the algorithm much flexibility.

Exploitation of the algorithm has still much more work to be done.