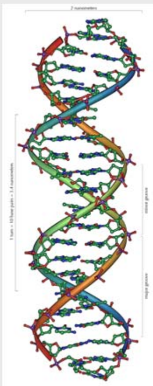
Genetic Algorithms:
Introduction and Principles

Marcus Schmitz
(Petru Eles)

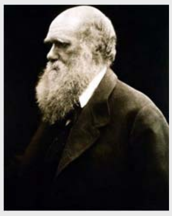
Outline

- Introduction
  - Origin
  - Jargon
- Basic Algorithm
- A GA Simulation by Hand
- Mathematical Foundation
- Implementation Issues
- Applications
  - Mapping
  - Traveling Salesman Problem



2

From Nature to Genetic Algorithms



- Charles R. Darwin (1809-1882)
  - The Origin of Species (1859)
    - “As natural selection works solely by and for the good of each being, all corporeal and mental endowments will tend to progress towards perfection.”
    - Survival of the fittest: Organisms that most fit to their environment will tend to survive the struggle for existence. Naturally, survivors pass on their hereditary dispositions to off-springs.

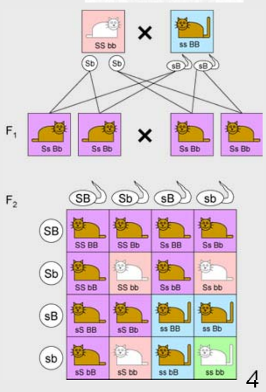
3

From Nature to Genetic Algorithms

- Gregor Mendel (1822-1884)

  Father of modern genetics

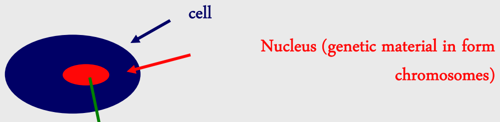
  - Mating experiments with pea plants
  - Mendel’s Laws
    - Law of Segregation
    - Law of Independent Assortment



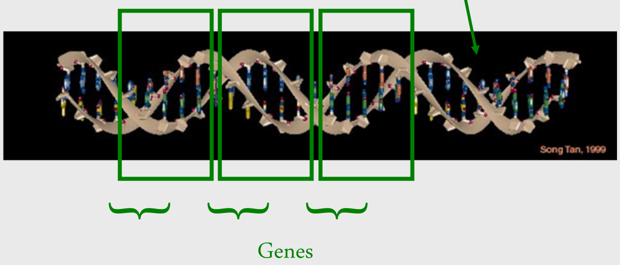
4

From Nature to Genetic Algorithms

- Reason for inheritance in organisms is the cell nucleus



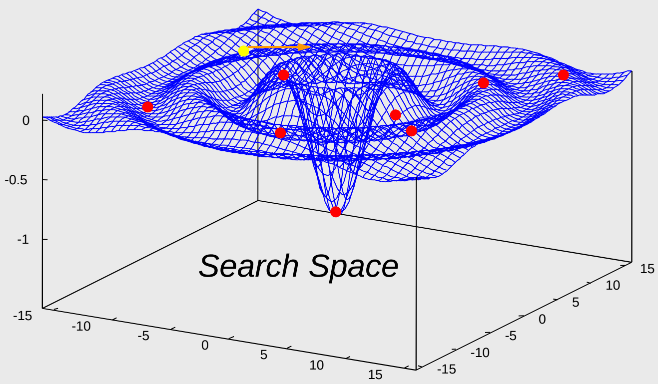
- Chromosome: long, continuous piece of DNA which carries genes



5

From Nature to Genetic Algorithms

- Genetic Algorithms (Rechenberg 1973)
  - Mimic the principles of natural selection to solve search and optimization problems



6

Introduction

- The algorithm requires feedback in form of a fitness value
  - Fitness function (Cost function)
    - Some idea of the solution quality to guide search
- Multiple objective optimization
- Multiple solutions are evolved in parallel
  - “Communication” through “building blocks” of solutions

7

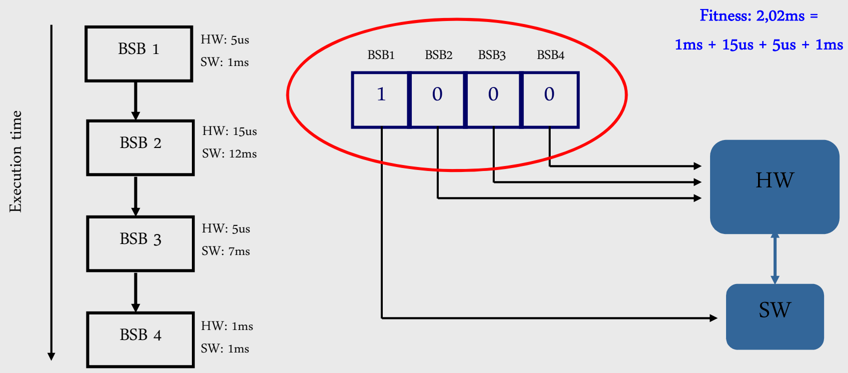
Jargon

- **Chromosome:** String of genes, representing a solution candidate
- **Population:** Set of chromosomes (possible solutions)
- **Gene:** Single entry in the chromosome, parameter of the solution set
- **Allele:** Value of a gene
- **Locus:** Gene position in the chromosome
- **Genetic operators:** Transform current chromosomes into new chromosomes

8

Jargon: Chromosome, Gene

- String of genes, representing a solution candidate
  - Example: HW/SW Co-Design



9

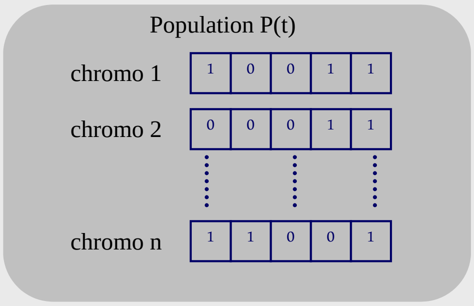
The Fundamental Algorithm

```
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination)
   begin
      t ← t + 1
      P(t) ← selection(P(t-1))
      crossover P(t)
      mutation P(t)
      evaluate P(t)
   end
end
```

10

Initialize Population

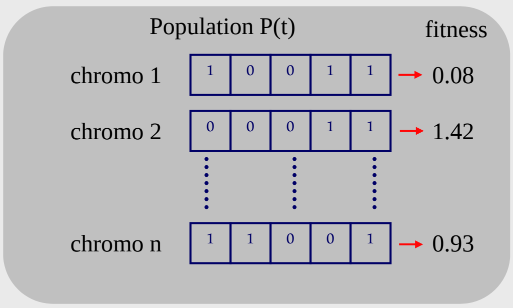
```
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination)
   begin
     t ← t + 1
     P(t) ← select(P(t-1))
     crossover P(t)
     mutation P(t)
     evaluate P(t)
   end
end
```



11

Evaluate Population

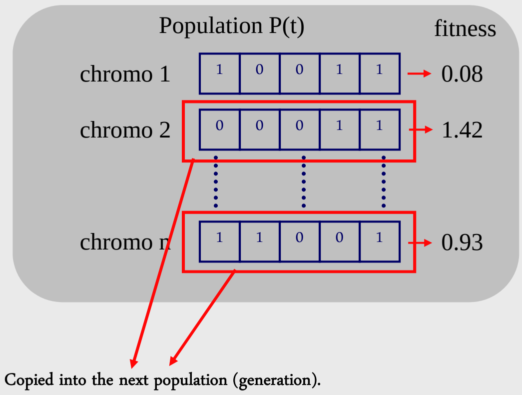
```
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination)
   begin
     t ← t + 1
     P(t) ← select(P(t-1))
     crossover P(t)
     mutation P(t)
     evaluate P(t)
   end
end
```



12

Selection

```
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination)
   begin
     t ← t + 1
     P(t) ← select(P(t-1))
     crossover P(t)
     mutation P(t)
     evaluate P(t)
   end
end
```



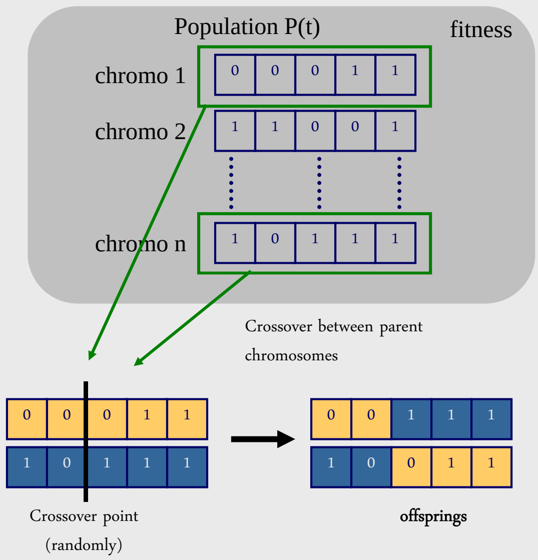
Selection is randomly performed, with a higher probability of selecting chromosomes of high fitness.

→ The number of individuals with high fitness increases from population to population

13

Crossover

```
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination)
   begin
     t ← t + 1
     P(t) ← select(P(t-1))
     crossover P(t)
     mutation P(t)
     evaluate P(t)
   end
end
```

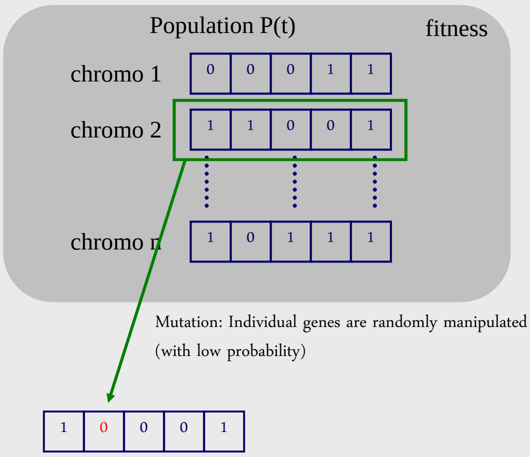


→ New solutions are generated from existing ones

14

Mutation

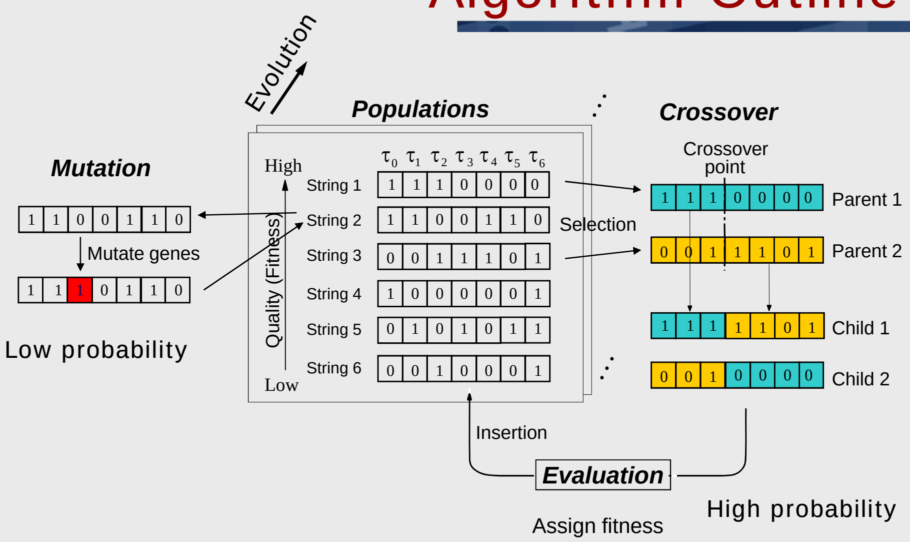
```
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination)
   begin
      t ← t + 1
      P(t) ← select(P(t-1))
      crossover P(t)
      mutation P(t)
      evaluate P(t)
   end
end
```



→ New individuals (points in the search space) are visited. Also solutions that would not be reached through crossover.

15

Algorithm Outline

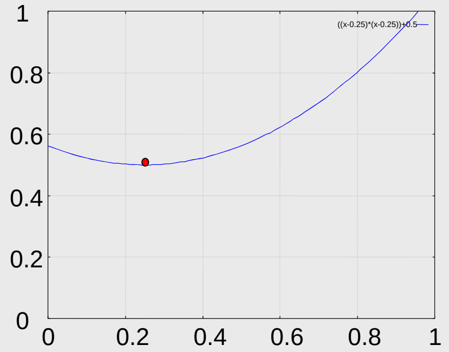


16

GA Simulation by Hand

$$f(x) = (x-0.25)^2 + 0.5$$

$$f(x_m) \le f(x), \forall x \in [0..1]$$ (find minimum)



Analytic solution:

$$f'(x) = 2x - 0.5$$

$$f'(x_m) = 0$$

$$x_m = 0.25$$

17

Chromosomes: Binary Encoding

- The interval [0..1] is encoded into a 8 bit string:

```
00000000 → 0
00000001 → 0.0039216
00000010 → 0.0078431
    ⋮       ⋮   ⋮
11111111 → 1
```

$$\frac{1-0}{2^8-1} = 0.0039216$$

18

Create Initial Population

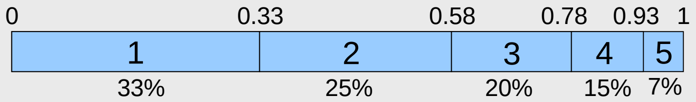
$P(t)$	x		Rank
00000011	→ 0.0117	$f(x) = 0.5568$	2
11011000	→ 0.8471	$f(x) = 0.8565$	5
01111111	→ 0.4980	$f(x) = 0.5615$	3
10001001	→ 0.5373	$f(x) = 0.5825$	4
00010010	→ 0.0706	$f(x) = 0.5322$	1

Fitness Function $f(x) = (x - 0.25)^2 + 0.5$

19

Selection

$P(t)$	x		Rank
00000011	→ 0.0117	$f(x) = 0.5568$	2
11011000	→ 0.8471	$f(x) = 0.8565$	5
01111111	→ 0.4980	$f(x) = 0.5615$	3
10001001	→ 0.5373	$f(x) = 0.5825$	4
00010010	→ 0.0706	$f(x) = 0.5322$	1



1. RandFloat(0,1) = 0.21 → 1
2. RandFloat(0,1) = 0.65 → 3
3. RandFloat(0,1) = 0.98 → 5

selected for P(t+1)

20

Crossover (2-point)

P(t+1)	*x*		*Rank*
	→ 0.0117	*f(x)* = 0.5568	2
11011000	→ 0.8471	*f(x)* = 0.8565	5
01111111	→ 0.4980	*f(x)* = 0.5615	3
	→ 0.5373	*f(x)* = 0.5825	4
00010010	→ 0.0706	*f(x)* = 0.5322	1

X-over at random point!

Parents: 01111111
00010010

		x	
Children:	01110010	→0.4471	*f(x)* = 0.5388
	00011111	→0.1216	*f(x)* = 0.5165

21

Replacement

P(t+1)	*x*		*Rank*
00000011	→ 0.0117	*f(x)* = 0.5568	2
11011000	→ 0.8471	*f(x)* = 0.8565	5
01111111	→ 0.4980	*f(x)* = 0.5615	3
10001001	→ 0.5373	*f(x)* = 0.5825	4
00010010	→ 0.0706	*f(x)* = 0.5322	1

Children:	01110010	→0.4471	*f(x)* = 0.5388
	00011111	→0.1216	*f(x)* = 0.5165

22

Second Iteration

$P(t+1)$	x		Rank
01110010	→ 0.4471	$f(x) = 0.5388$	3
11011000	→ 0.8471	$f(x) = 0.8565$	5
01111111	→ 0.4980	$f(x) = 0.5615$	4
00011111	→ 0.1216	$f(x) = 0.5165$	1
00010010	→ 0.0706	$f(x) = 0.5322$	2

- Next selection for crossover: 1 and 4

01|111111
00|011111

01|011111 →0.3755 $f(x) = 0.5158$
00|111111 →0.2471 $f(x) = \mathbf{0.50001}$

23

Why do GAs work?

		Rank
00000011	$f(x) = 0.5568$	3
11011000	$f(x) = 0.8565$	5
01111111	$f(x) = 0.5615$	4
00011111	$f(x) = 0.5165$	1
00010010	$f(x) = 0.5322$	2

- *Relationship between similarities and high fitness!*
  - Information to help guide the search

24

Similarity Templates (Schemata)

- Which information is admitted?
  - Schemata help to answer this question

***0000** matches **{00000,10000}**

111 matches **{01110,01110, 11110,11111}**

***** ← Don't care symbol

k^l: alternative string ($2^5 = 32$)

$(k+1)^l$: schemata ($3^5 = 243$)

25

Information Amount

- Number of unique schemata in population

  Each string is a member of 2^l schemata
  - Between 2^l and $n \cdot 2^l$ (n: population size)

- Defining length of a schema

  Distance between last and first fixed string position

  d(*11*00*) = 6 – 2 = 4

- Order of a schema

  Number of 0 and 1 (fixed) positions

  O(*11*00*) = 4

26

Usefully Processed?

- Effect of Selection (Reproduction)
  - Ever-increasing number of individuals with good similarity patterns
- Effect of Crossover
  - Schema can be disrupted or left unscathed

    Examples: **1***0** and ****11***
- Effect of Mutation
  - Schema is disrupted with low frequently (low mutation rate)
- Conclusion: Highly fit schemata with short-defining-length and low order (*building blocks*) are propagated from generation to generation.

27

Algorithm Setup & Parameters

- Chromosome type (Encoding)
- Population type & size
- Selection scheme
- Crossover types (2-point, 3-point, etc.)
- Mutation strategy & probability
- Fitness function
- Termination criterion

28

Chromosome

- Principle of meaningful building blocks
  - “Select encoding so that short, low-order schemata are relevant to the underlying problem”, i.e., short distance between related bit positions
- Principle of minimal alphabets
  - “Choose smallest alphabet that permits a natural expression of the problem”

29

Population Types & Size

- Generation-based GAs
  - In each generation all individuals of the population are replaced
- Steady-state GAs
  - Generational overlap: A certain fraction of the population is replaced by new individuals
- Multiple Populations with Immigration
  - Several populations evolve in parallel, individuals can immigrate between population islands (computing clusters)
- Typical Sizes 25 - 2000 chromosomes

30

Initial Population

- Randomly selected individuals
- Mixed population
  - A fixed amount of individual constructed through different constructive heuristic
  - In addition, random individuals

31

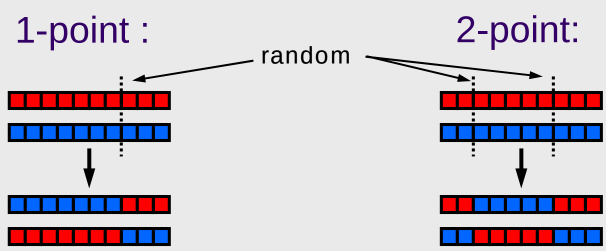
Selection Scheme

Assignment of reproduction opportunities to the individuals

- Roulette Wheel Selection
  - Fitness determines selection probability
- Ranking-based Selection
  - Ranking determines selection probability
  - Avoids problems with “super-individuals”
- Tournament Selection
  - Randomly select two individuals, the better one is chosen

32

Crossover Types



33

String Encoding

- Recall: Short defining-length, low order, high fitness schemata (building blocks) recombine

→ The coding decision influences the efficiency of GAs

```
a b c d e f
1 * * * * 1
```

Likely to be disrupted (long defining-length) → 1 * * * * 1 → highest average fitness

Reordering of genes

```
a f c d e b
1 1 * * * *
```

Likely to be left undisrupted (short defining-length) → 1 1 * * * *

34

Mutation Strategies & Probability

- Constant Mutation Rate
  - Genes are altered permanently during optimization with fixed probability (common value <1%)
- Decreasing Mutation Rate
  - An initially high mutation rate decreases during optimization run
- Stimulating Mutation
  - If premature convergence is detected, an increasing number of individuals are mutated

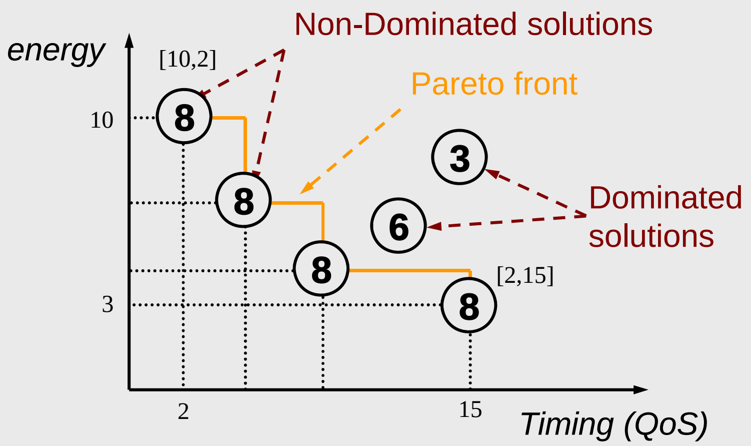
35

Fitness function

- Single-objective optimization
  - Fitness depends on calculated cost
- Multi-objective optimization
  - Objective weighting: $F(\mathbf{x}) = \sum_{i=1}^{k} w_i \cdot f_i(\mathbf{x})$
  - Pareto ranking: Distance based

36

Pareto Ranking



Non-dominated solutions: at least on of the solution weights is the smallest among all other solutions!

37

Termination Criterion

- A given maximal number of generations has been reached
- A certain amount of generations has not produced any further improvements
- The diversity in the population has reached a lower limit

38

Applicability

- Large Search Space
  - Not perfectly smooth (no gradient-based tech.)
  - Not unimodal (extreme points)
  - Not well understood
  - Noisy fitness function
- Global optimum is not essential
  - High quality solution is sufficient

39

Knowledge-based Techniques

- In the most general case, GAs are “blind” heuristics, *i.e.*, no problem specific knowledge is required
- Hybrid Schemes
  - Example: GA + local search (GA finds hills, local search climbs hills)
  - Performance improvement

40

Evolution Programs

- Difference between GAs and EPs?
  - GAs: binary string representations
  - EPs: Complex data structures
  - GAs: Standard genetic operators
  - EPs: Specialized genetic operators

41

Available Implementations

- GALib (MIT, http://lancet.mit.edu/ga)
  - Includes several GA types
  - Comes with numerous crossover, replacement, mutation types
  - Easily adaptable to specific problems (new genetic operators can be created)
- GAUL (GNU, http://gaul.sourceforge.net)
  - Support for multiple, simultaneously evolving populations (computing clusters)
  - Additional optimization algorithms are built-in
    - Simulated annealing
    - Tabu search

42

Further Readings

- Books
  - Goldberg, “Genetic Algorithms in Search, Optimization & Machine Learning”
  - Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs”
  - Mazumder and Rudnick, “Genetic Algorithms for VLSI Design, Layout & Test Automation”
- Conference proceedings
  - International Conference on Genetic Algorithms
  - International Conference on Evolutionary Programming
- Journals
  - IEEE Transactions on Evolutionary Computation
  - Evolutionary Computation Journal (MIT Press)

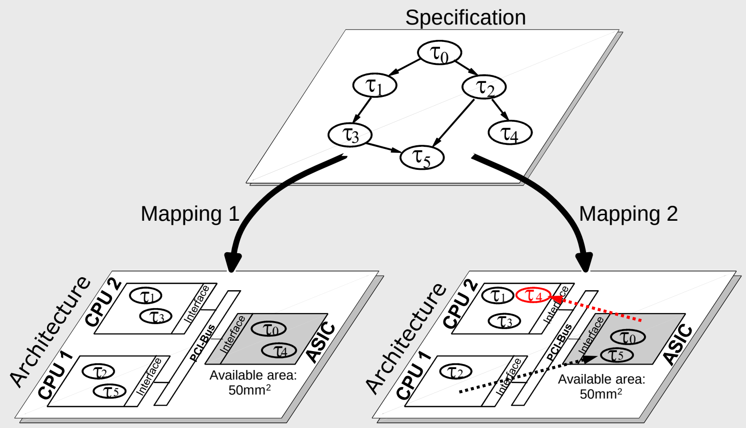
43

Applications

- Application Mapping in Multiprocessor Systems
- Traveling Salesman Problem

44

Application Mapping



Performance: 10*ms*
Power dissipation: 350*mW*

Performance: 9*ms*
Power dissipation: 370*mW*

45

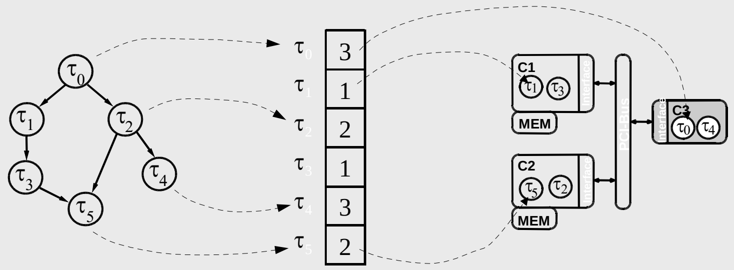
Task Properties

- $t_i(C)$ is the execution time of task τ_i on component C
- $a_i(C)$ is the area required to accommodate task τ_i on component C
- $P_i(C)$ is the power dissipated by task τ_i on component C

- Competing objectives:
  - Performance
  - Area
  - Power consumption

46

Encoding: Mapping String



Task Graph **String** **Architecture**

- Locus determines task position
- Allele determines task mapping

47

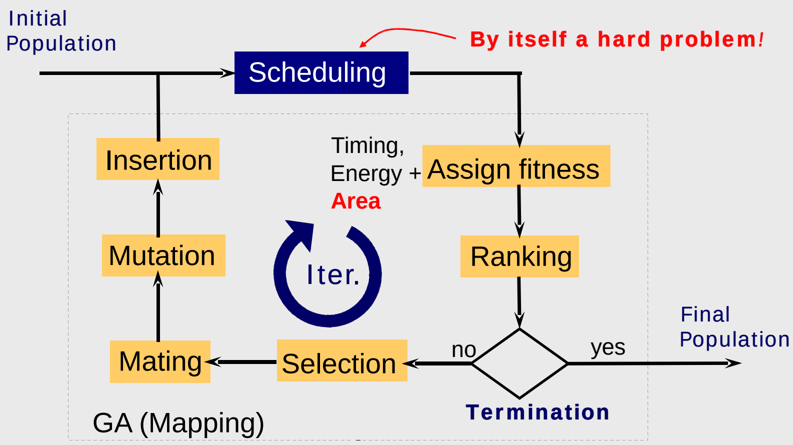
Fitness Function

$$F_M = \underbrace{\sum_{\tau \in T} E(\tau)}_{energy} \cdot \underbrace{\left(1 + \frac{DV^2}{T_{rep}^2}\right)}_{pentalty\_time} \cdot \underbrace{\prod_{\pi \in C} AP_\pi}_{penalty\_area}$$

$$AP_\pi = \begin{cases} 1 & \text{if } AA_\pi \geq SA_\pi \\ k \cdot \left(\frac{UA_\pi}{AA_\pi} - 1\right) + 1 & \text{otherwise} \end{cases}$$

48

GA Mapping Algorithm



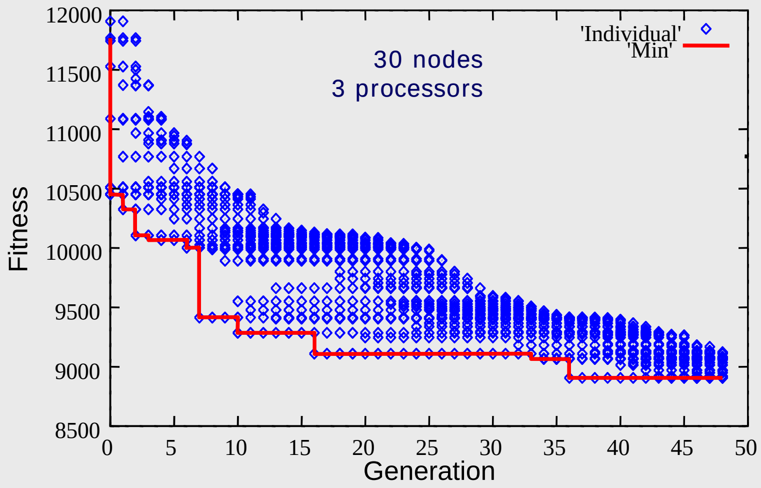
49

Experimental Setup

- Population size: 50
- Generational overlap: 20%
- Two-point crossover
- Dynamic mutation probability 5%

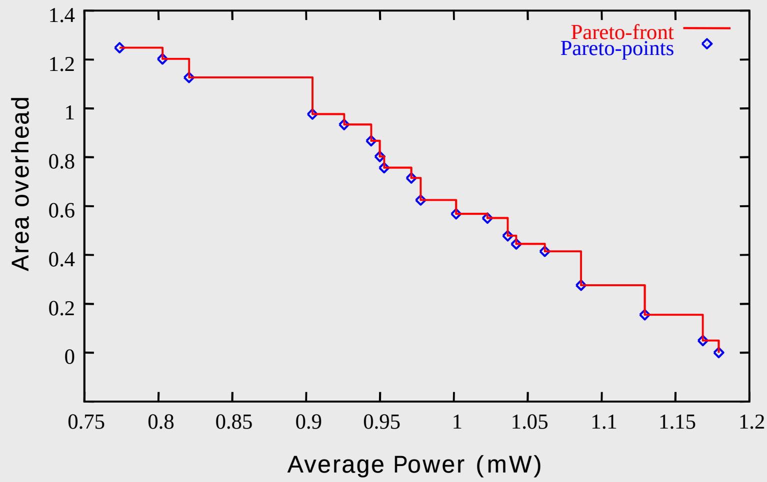
50

Evolution Run



51

Multi-Objective Optimization



52

Experimental Results

No. Nodes	CPU time (s)
20	5.5
30	11
40	37
100	127

PentiumIII/500MHz

Optimization times include overheads due to scheduling and energy management

53

Traveling Salesman Problem

- "...given a finite number *n* of "cities" along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point."
- The problem has a solution space of $(n-1)!/2$

Cities	Possible routes
10	181,440
25	310e21
100	466e153

54

Recombination Problem

- Standard GA operators fail to produces meaningful chromosomes
  - Example: 1-point crossover

```
[A B C | D E F]          [A B C A E F]
                  ==>
[B D C | A E F]          [B D C D E F]
```

NOT VALID!

- Repair algorithm to restore a valid solution is not effective
- Using appropriate operator that lead to feasible solutions

55

Edge Recombination Operator

- Similarities between tours should be preserved
  - Offspring should be constructed from “links” that exist in the parent tours
- Key to solve the problem is a meaningful recombination technique
  - For example: Edge recombination operator [1]

56

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: B F C E	**D: C E B**
B: A C D F	**E: D F A**
C: B D A	**F: A E B**

Child tour: **[? ? ? ? ? ?]**

57

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: B F C E	**D: C E B**
B: A C D F	**E: D F A**
C: B D A	**F: A E B**

1. Initialize child tour with one of the two initial cities of the parents.

   Randomly chosen B.

Child tour: **[B ? ? ? ? ?]**

58

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F C E	**D: C E ~~B~~**
B: A C D F	**E: D F A**
C: ~~B~~ D A	**F: A E ~~B~~**

2. Remove all occurrences of B in the edge map.

Child tour: **[B ? ? ? ? ?]**

59

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F C E	**D: C E ~~B~~**
B: A C D F	**E: D F A**
C: ~~B~~ D A	**F: A E ~~B~~**

3. Which of the cities in edge list B has the fewest cities in its own edge list? C, D, F!

Randomly chosen C.

Child tour: **[B C ? ? ? ?]**

60

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ E	**D: ~~C~~ E ~~B~~**
B: A ~~C~~ D F	**E: D F A**
C: ~~B~~ D A	**F: A E ~~B~~**

4. Remove all occurrences of C in the edge lists.

Child tour: **[B C ? ? ? ?]**

61

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ E	**D: ~~C~~ E ~~B~~**
B: A ~~C~~ D F	**E: D F A**
C: ~~B~~ D A	**F: A E ~~B~~**

5. Which of the cities in edge list C has the fewest cities in its own edge list? D!

   Chosen D.

Child tour: **[B C D ? ? ?]**

62

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ E	**D: ~~C~~ E ~~B~~**
B: A ~~C~~ ~~D~~ F	**E: ~~D~~ F A**
C: ~~B~~ ~~D~~ A	**F: A E ~~B~~**

6. Remove all occurrences of D in the edge lists.

Child tour: **[B C D ? ? ?]**

63

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ E	**D: ~~C~~ E ~~B~~**
B: A ~~C~~ ~~D~~ F	**E: ~~D~~ F A**
C: ~~B~~ ~~D~~ A	**F: A E ~~B~~**

7. Which of the cities in edge list D has the fewest cities in its own edge list? E!

   Chosen E.

Child tour: **[B C D E ? ?]**

64

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ ~~E~~
B: A ~~C~~ ~~D~~ F
C: ~~B~~ ~~D~~ A
D: ~~C~~ ~~E~~ ~~B~~
E: ~~D~~ F A
F: A ~~E~~ ~~B~~

8. Remove all occurrences of E in the edge lists.

Child tour: **[B C D E ? ?]**

65

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ ~~E~~
B: A ~~C~~ ~~D~~ F
C: ~~B~~ ~~D~~ A
D: ~~C~~ ~~E~~ ~~B~~
E: ~~D~~ F A
F: A ~~E~~ ~~B~~

9. Which of the cities in edge list E has the fewest cities in its own edge list? F!

   Randomly chosen A.

Child tour: **[B C D E A ?]**

66

An Example

- Parent tours: **[A B C D E F]** and **[B D C A E F]**
- Edge map:

A: ~~B~~ F ~~C~~ E	**D: ~~C~~ E ~~B~~**
B: A ~~C~~ D F	**E: D F A**
C: ~~B~~ D A	**F: A E ~~B~~**

All cities have been visit → STOP

Child tour: **[B C D E A F]**

67

GA-TSP: Results

- 30 cities (optimal solution 420)
  - 4.42e30 possible tours
  - 10 sub-populations with a size of 200 each
  - 7,000 recombinations
  - 30 out of 30 runs optimal solution found
- 105 cities (optimal solution 14,383)
  - 5.14e165 possible tours
  - 10 sub-populations with a size of 1000 each
  - 200,000 recombinations
  - 15 out of 30 runs optimal solution found
  - 15 out of 30 runs with 1 percent of optimal solution

68

References

[1] D. Whitley et al, "The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination", 1993.

69

Conclusions

- Simple GA has been introduced
- We have examined how GAs work
- Implementation issues
  - Crossovers
  - Encoding
- Applications
  - Task mapping
  - TSP
- GAs provide a robust, easy to implement heuristic search strategy that can be applied to large number of optimization and search problems

70