Optimal Data Base Design Problem

Group 1

Luigi Amato, Andrea Attili, Valentina Avigliano,

Enzo Bisceglia, Andrea Costa, Michele D'Amico, Giada Morat

Why a Memetic algorithm?

- We have chosen to implement a Memetic Algorithm, a Genetic Algorithm integrated with Local Search;
- Very efficient in exploring the broad solution space;
- Suitable for parallel computation;
- It enhances randomization;
- Time is the stopping condition.

Chromosome structure

- Our population is formed by 200 possible solutions (chromosomes) always feasible;
- Each chromosome is an array, as long as the number of queries;
- Each gene contains the configuration used to satisfy the corresponding query;
- In order not to exceed the available memory, we activate only some genes;
- Genes are evaluated in a random order, and they are activated until memory is still available.

Example

Memory available = 30

4 (10)	5 (20)	23 (6)	35 (11)	80 (17)
5	3	1	2	4

- 5 genes for 5 queries
- 3rd and 4th genes are activated first
- 2nd and 5th genes are not activate (not enough memory left!)
- 1st gene is activated

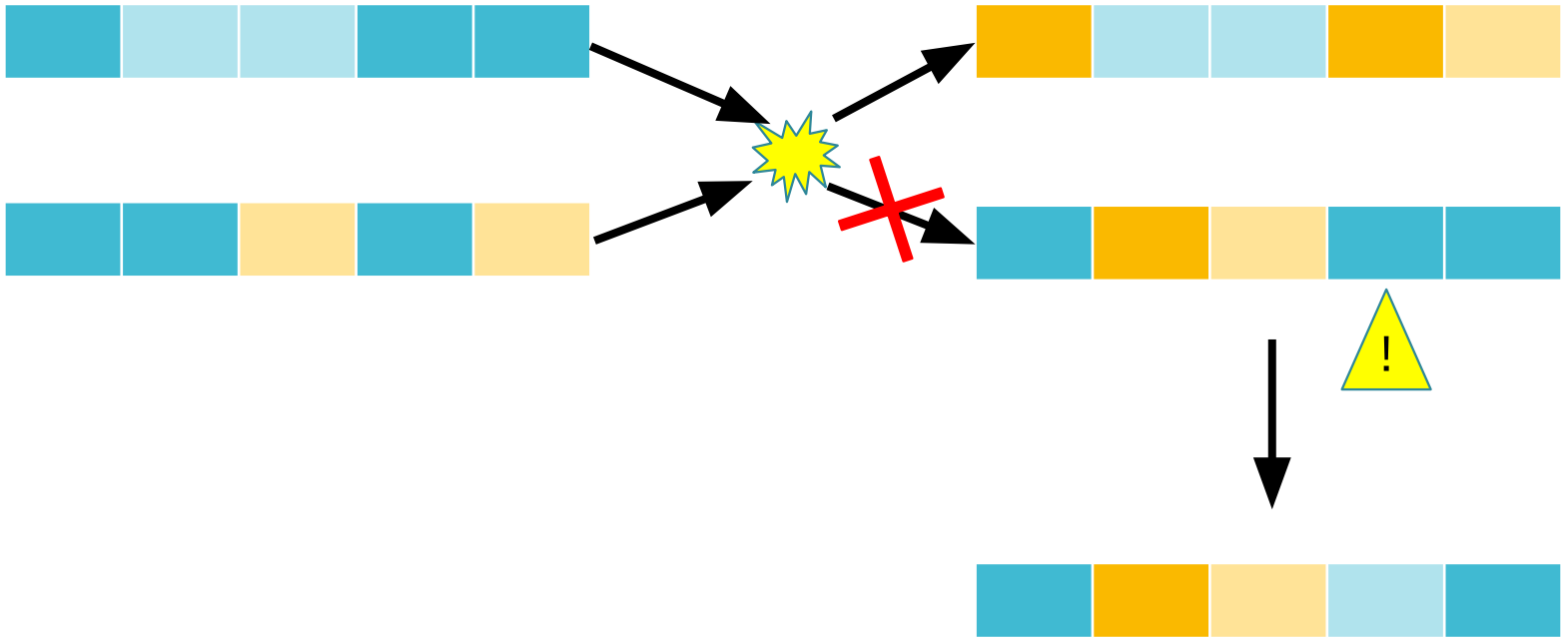
The initial population is generated randomly.

Generation of the new population

1. *Selection* (30% of the times):
   - 4 elements are selected randomly;
   - the one with the highest fitness is inserted into the new population;
   - the process is repeated twice;
   - we obtain two chromosomes for the new population.
2. *Crossover* (70% of the times):
   - two parents are randomly selected;
   - if they are different, they generate two children.

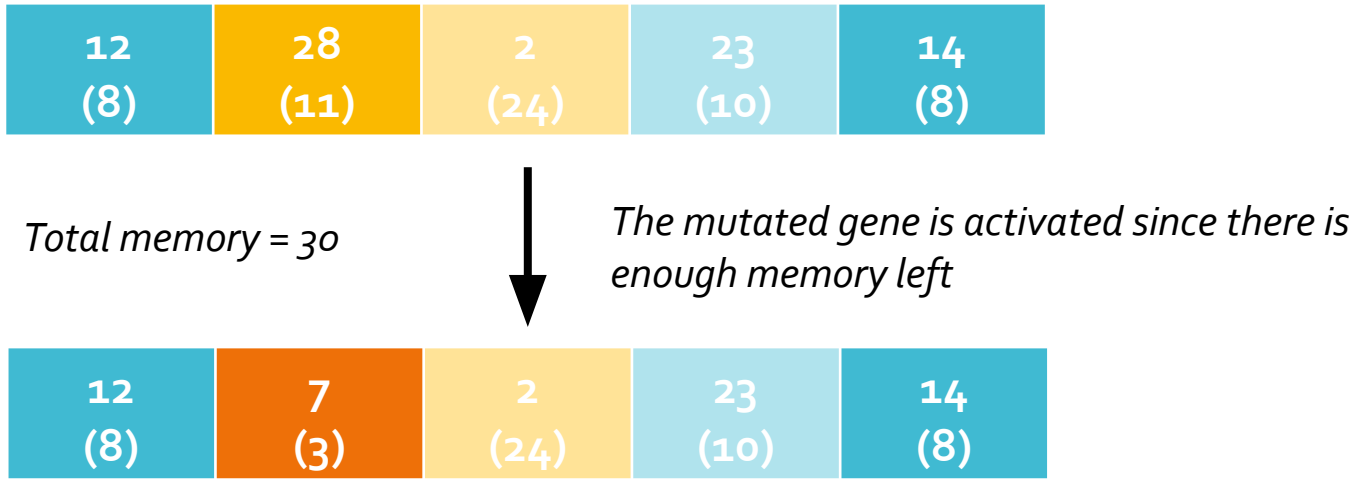
Crossover

- If the first child inherits a gene from the father, the second one inherits the same gene from the mother, and viceversa;
- Active / inactive genes are passed on, until there is enough memory left.



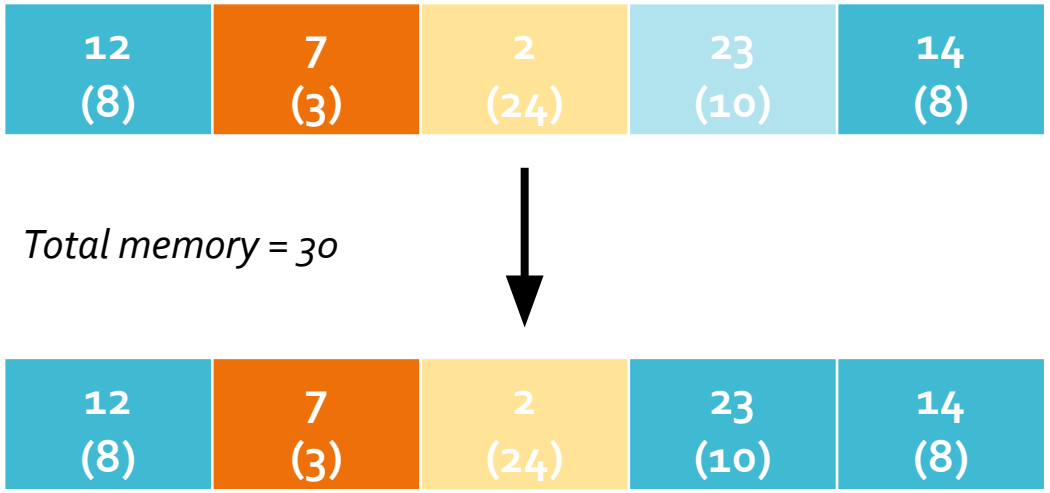
Crossover. Mutation 1

- It's performed with probability 10/|Q|;
- A gene is selected randomly, not depending on its status (activated or not);
- The configuration associated to this gene is changed;
- If there's enough memory available for the new configuration, the gene is activated.



Crossover. Mutation 2

- It's performed with probability 10/|Q|;
- An inactive gene is chosen randomly: if there's enough memory left, it's activated.



Fill the green cells only!!! Do not change anything else!!!

	Indexes	Queries	Configurations	Benchmark (obj.f.)	Opt	Your algorithm best solution obj.f.	Gap %
instance01.odbdp	50	50	500	5951	YES	5699,000	4,23
instance02.odbdp	50	100	500	1513	YES	1213,000	19,83
instance03.odbdp	50	50	1000	7484	YES	7334,000	2,00
instance04.odbdp	50	100	500	22010	YES	19461,000	11,58
instance05.odbdp	50	50	500	13874	YES	12604,000	9,15
instance06.odbdp	50	50	1000	7003	YES	5995,000	14,39
instance07.odbdp	100	100	1000	73772	YES	72149,000	2,20
instance08.odbdp	100	100	1000	32999	YES	31934,000	3,23
instance09.odbdp	50	50	500	1710	YES	1361,000	20,41
instance10.odbdp	50	50	500	1672	YES	1226,000	26,67
instance11.odbdp	100	100	500	10749	YES	10418,000	3,08
instance12.odbdp	50	100	1000	2287	YES	1510,000	33,97
instance13.odbdp	50	100	1000	26938	NO	23263,000	13,64
instance14.odbdp	100	50	500	29280	NO	28062,000	4,16
instance15.odbdp	50	100	1000	12351	NO	9955,000	19,40
instance16.odbdp	100	50	1000	14110	NO	13965,000	1,03
instance17.odbdp	50	100	1000	12218	NO	10342,000	15,35
instance18.odbdp	50	50	1000	2081	NO	1568,000	24,65
instance19.odbdp	100	50	1000	4257	NO	3575,000	16,02
instance20.odbdp	100	100	500	3406	NO	1469,000	56,87
						Best:	1,03
						Avg:	15,09
						Worst:	56,87

Local Search

Previous Results without Local Search

Local Search

- Based on configuration neighborhood.
- Performed on both children chromosome separately.

Procedure:

1. **Scanning of the indices** used by the active configurations of the chromosome;
2. Creation of a **configuration neighborhood** which contains all the configurations (including the ones we are not currently using) that use the same indices of the active configurations;
3. In random order and for every gene, **swapping** of the current configuration with one of the neighbor configurations in order to improve the fitness.

Local Search. Example

12 (8)	7 (3)	2 (24)	23 (10)	14 (8)

- Indices in common: 127, 52, 23, 26, 275;
- Configuration Neighborhood: 12, 7, 23, **13, 27.**

12 (8)	7 (3)	12 (8)	23 (10)	14 (8)
12 (8)	7 (3)	7 (3)	23 (10)	14 (8)
12 (8)	7 (3)	23 (10)	23 (10)	14 (8)
12 (8)	7 (3)	13 (10)	23 (10)	14 (8)
12 (8)	7 (3)	27 (5)	23 (10)	14 (8)

The solution with the best fitness is the following:

12 (8)	7 (3)	27 (5)	23 (10)	14 (8)

This is one of the final children chromosome.
We pass to the new generation the best children.

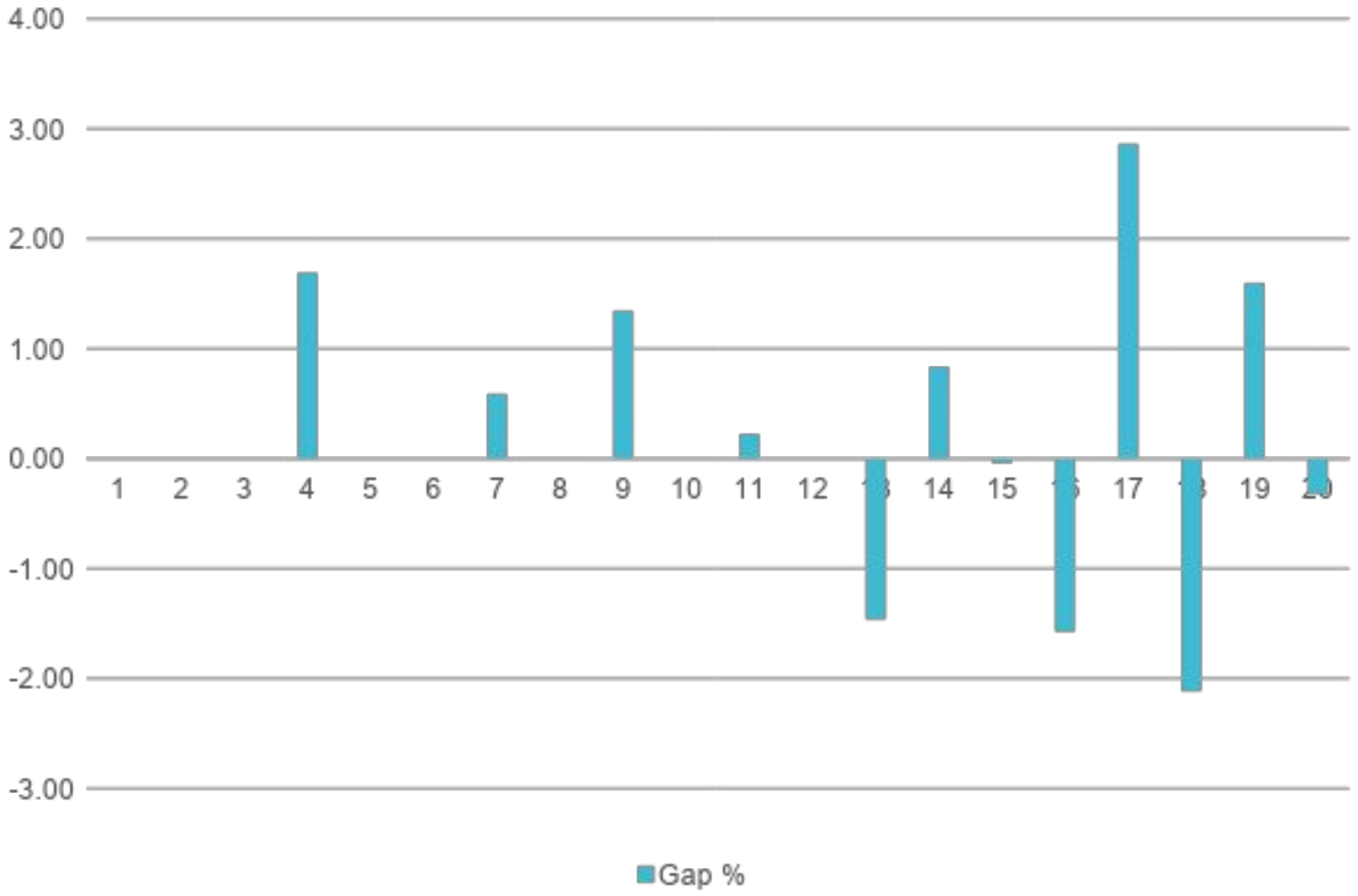
Selection of the optimal solution

- The chromosome with the highest fitness among the whole population will be the current optimal solution
- Genes corresponding to configurations that don't give a positive net gain are inactivated

Best final results

	Indexes	Queries	Configurations	Benchmark (obj.f.)	Opt	Your algorithm best solution obj.f.	Gap %
instance01.odbdp	50	50	500	5951	YES	5951,000	0,00
instance02.odbdp	50	100	500	1513	YES	1513,000	0,00
instance03.odbdp	50	50	1000	7484	YES	7484,000	0,00
instance04.odbdp	50	100	500	22010	YES	21639,000	1,69
instance05.odbdp	50	50	500	13874	YES	13874,000	0,00
instance06.odbdp	50	50	1000	7003	YES	7003,000	0,00
instance07.odbdp	100	100	1000	73772	YES	73338,000	0,59
instance08.odbdp	100	100	1000	32999	YES	32999,000	0,00
instance09.odbdp	50	50	500	1710	YES	1687,000	1,35
instance10.odbdp	50	50	500	1672	YES	1672,000	0,00
instance11.odbdp	100	100	500	10749	YES	10725,000	0,22
instance12.odbdp	50	100	1000	2287	YES	2287,000	0,00
instance13.odbdp	50	100	1000	26938	NO	27332,000	-1,46
instance14.odbdp	100	50	500	29280	NO	29035,000	0,84
instance15.odbdp	50	100	1000	12351	NO	12356,000	-0,04
instance16.odbdp	100	50	1000	14110	NO	14332,000	-1,57
instance17.odbdp	50	100	1000	12218	NO	11868,000	2,86
instance18.odbdp	50	50	1000	2081	NO	2125,000	-2,11
instance19.odbdp	100	50	1000	4257	NO	4189,000	1,60
instance20.odbdp	100	100	500	3406	NO	3417,000	-0,32
						Best:	-2,11
						Avg:	0,18
						Worst:	2,86

Final Results



Final considerations

Possible Improvements:

- Avoiding Local Optima;
- We didn’t pay much attention to the strong domain shrinking heuristics;
- Tuning Improvement;
- Index-based crossover;
- Local Search Improvement.