Genetic Algorithms and Evolutionary Computing: Exercise session 1

2020-2021

1 Simple Genetic Algorithm

The 'Simple Genetic Algorithm' (as described by Goldberg) uses a representation by bitstrings, roulette wheel selection, one-point crossover (consists of exchanging all of the bits after a randomly-chosen crossover point at the same location in both strings) with probability $p_c = 100\%$, bitwise mutation with probability p_m, generational survivor selection ($\lambda = \mu$). Consider an initial population that consists of four individuals (bitstrings of length 6):

$$v_1 = 001100$$

$$v_2 = 001001$$

$$v_3 = 011011$$

$$v_3 = 100110$$

Compute manually two consecutive generations according to the simple genetic algorithm. First assume probability of mutation $p_m = 0$. Afterwards recalculate the first generation using $p_m = 0.1$. The individuals, the associated phenotypes and fitness values are given in Table. 1 Use the pseudo-random numbers given in Table. 2 and Table. 3 when necessary (always use the first unused number). For every generation, observe the maximal and average fitness in the population.

1 Do the chromosomes with a fitness value above average always produce a better offspring?

2 Is the crossover operator able to fully explore the search space? What are the limitations of this exploration?

3 Can you identify any specific patterns in the final population? Do you observe similarities in the genotypes or phenotypes?

4 Does the probability of mutation plays an important role?

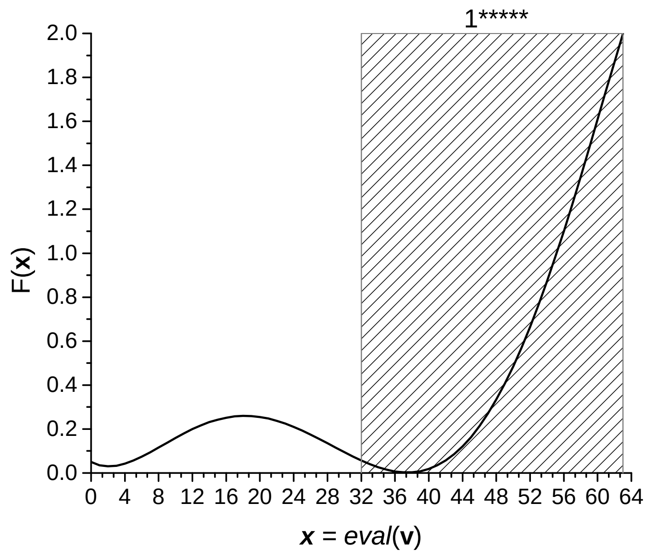

Figure 1: Plot of the fitness values given in Table 1

Table 1: Mapping between genotype, phenotype and values of fitness function

Genotype **v**	Phenotype **x**	Fitness $F(\mathbf{x})$	Genotype **v**	Phenotype **x**	Fitness $F(\mathbf{x})$
000000	0	0.050	100000	32	0.057
000001	1	0.035	100001	33	0.040
000010	2	0.030	100010	34	0.026
000011	3	0.033	100011	35	0.015
000100	4	0.042	100100	36	0.007
000101	5	0.056	100101	37	0.003
000110	6	0.074	**100110**	38	0.003
000111	7	0.094	100111	39	0.008
001000	8	0.116	101000	40	0.019
001001	9	0.138	101001	41	0.035
001010	10	0.160	101010	42	0.057
001011	11	0.180	101011	43	0.085
001100	12	0.200	101100	44	0.121
001101	13	0.217	101101	45	0.163
001110	14	0.231	101110	46	0.213
001111	15	0.243	101111	47	0.270
010000	16	0.252	110000	48	0.334
010001	17	0.257	110001	49	0.405
010010	18	0.260	110010	50	0.485
010011	19	0.259	110011	51	0.571
010100	20	0.255	110100	52	0.664
010101	21	0.248	110101	53	0.764
010110	22	0.238	110110	54	0.871
010111	23	0.225	110111	55	0.983
011000	24	0.210	111000	56	1.100
011001	25	0.193	111001	57	1.222
011010	26	0.175	111010	58	1.348
011011	27	0.156	111011	59	1.477
011100	28	0.135	111100	60	1.608
011101	29	0.115	111101	61	1.739
011110	30	0.094	111110	62	1.870
011111	31	0.075	111111	63	1.999

Table 2: Uniformly distributed integer pseudo-random numbers within range $[1, 5]$ (read row-wise)

4	2	3	1	4	1	2	3
5	2	1	3	1	5	4	2
1	2	4	2	2	4	3	5
3	1	3	4	3	1	5	3

Table 3: Uniformly distributed pseudo-random numbers (read row-wise)

0.0605	0.6280	0.1672	0.7395	0.2691	0.9831	0.6981	0.1711
0.3993	0.2920	0.1062	0.9516	0.4228	0.3015	0.6665	0.0326
0.8469	0.4317	0.3724	0.9203	0.5479	0.7011	0.1781	0.5612
0.4168	0.0155	0.1981	0.0527	0.9427	0.6663	0.1280	0.8819
0.6569	0.9841	0.4897	0.7379	0.4177	0.5391	0.9991	0.6692

2 Adjacency representation of TSP: "(cycle notation)"

Please see [Eiben] and the slides on the TSP presented during the lectures.

Exercise 1: Evolutionary operators for the adjacency representation

Let's consider two tours shown in the first row of Fig. 2 and denoted as Parent 1 and Parent 2, respectively.

- Write the adjacency representation of these tours.
- Create a pair of offspring by applying the alternating edge crossover operator. Draw the resulting tours in the second row of Fig. 2.
- Modify each parent individually by means of the swap (=reciprocal exchange) mutation operator. Draw the tours corresponding to the mutated parent individuals in the third row of Fig. 2.
- Answer the following questions:
  1. Are the parent individuals similar? How to quantify this resemblance?
  2. How many edges are preserved in the offspring if the alternating edge crossover operator is used?
  3. How many edges are kept unmodified by the swap (=reciprocal exchange) operator?

Exercise 2: Distance between individuals

Sometimes it is useful to calculate the similarity of the genotype of two individuals. For instance, this can help to assess the diversity of a population. Propose some methods that can quantify the *distance* between the genotype of two individuals in case of

1. binary coding, and
2. adjacency and path representation for TSP. Analyse the distances between the tours (given here in path representation): t_1 = `1-2-3-4-5`, t_2 = `5-4-1-3-2`, t_3 = `3-4-5-1-2`, t_4 = `3-2-1-5-4` and t_5 = `4-2-5-3-1`.

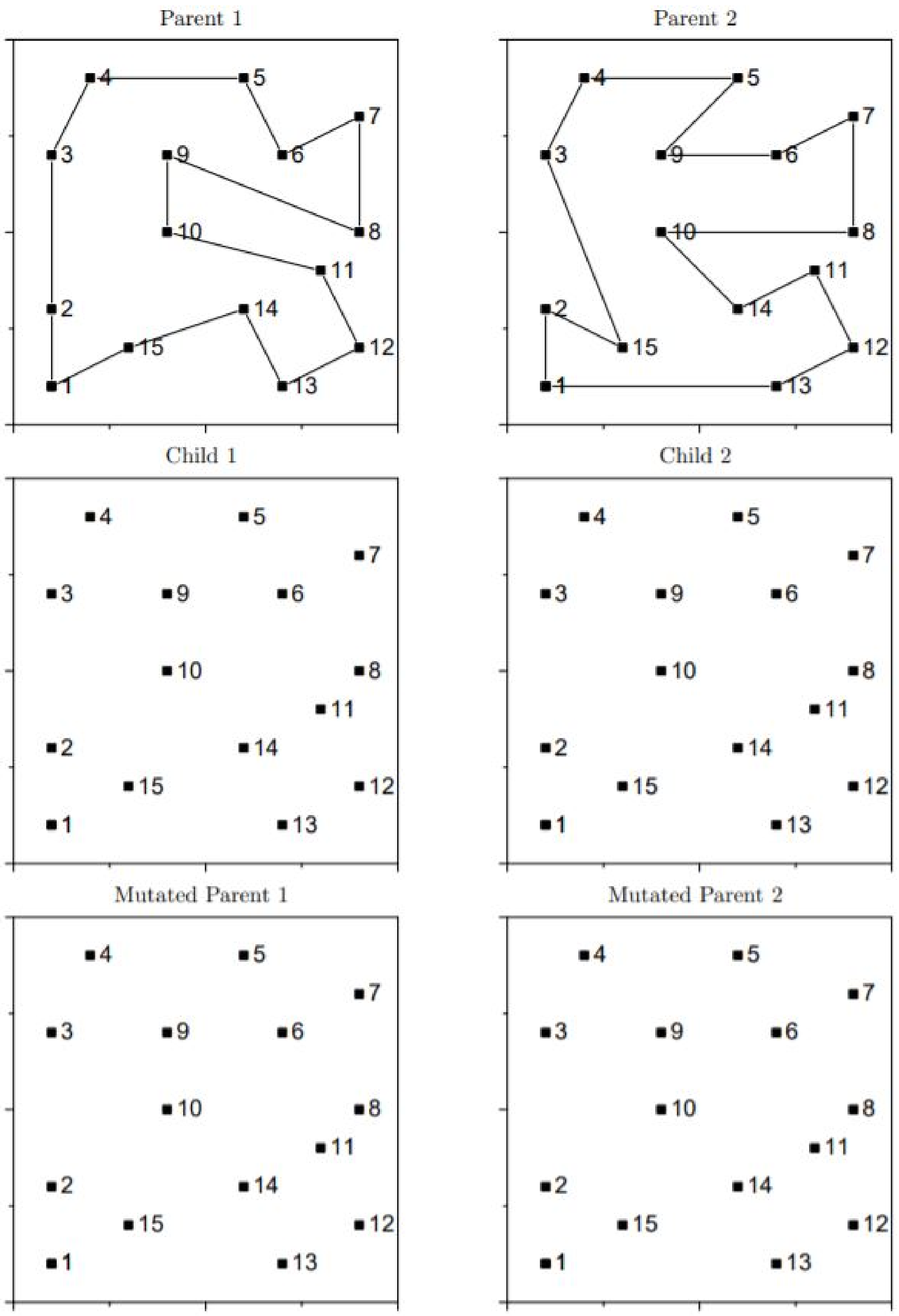

Figure 2: Caption

3 Timetabling

The timetabling problem: the problem of scheduling lectures, practical sessions, examinations, etc. taking into account:

1. Hard Constraints, e.g.: a teacher can only teach one lecture in a given timeslot
2. Soft Constraints, e.g.: a teacher should not teach the same class (student group) in consecutive timeslots.

There are several variants of this problem depending on the 'data' and the constraints.

University Course Timetabling problem

Given: sets of teachers, time slots, classes (student groups), sets of lectures to be given by a certain teacher to a certain class, ...
Hard Constraints:

1. no teacher or class is involved in more than one lecture in a given time slot
2. tables with availabilities of teachers and classes in each time slot

Read the papers [1], [2] and check the methods in these papers for UCTP and solve the tasks below

Task 1

Prepare a list of hard constraints and soft constraints for the UCT problem that are applicable at KU Leuven.

Task 2

Construct a fitness function $f(t)$ (t =timetable) for the lecture timetabling which takes into account (i.e. penalizes) the hard constraints C_i, $i = 1, \dots, l$, such that $f(t) = 1$ if no constraints are violated and $f(t)$ decreases to 0 if the number of violated constraints increases.
Modify the fitness function to take the soft constraints into account as well.

Task3

Find or propose another representation of a timetable that can be used in the UTCP.

Task4

Propose appropriate genetic operators (mutation, crossover) for that representation of the timetable.

References

[1] Peter Ross, Dave Corne, and Hsiao-Lan Fang. Successful lecture timetabling with evolutionary algorithms. In *Proceedings of the ECAI*, volume 94, 1994.

[2] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Metaheuristics for high school timetabling. *Computational optimization and applications*, 9(3):275–298, 1998.