Отчет

Выполнила: Залесская Галина

Генетический алгоритм подбирал параметры для задачи biclustering. Для каждого инстанса было сменено 5 поколений. Это немного, но очень близкое к оптимуму решение находится почти сразу и потом почти не меняется (именно из-за своей близости к оптимуму).
В таблице показано, как меняется значение лучшего решения в зависимости от номера поколения.

	0	1	2	3	4
20x20	0.430556	0.430556	0.430556	0.430556	0.434483
24x40	0.461538	0.461538	0.461538	0.461538	0.461538
30x50	0.505208	0.505319	0.505319	0.505319	0.505319
30x90	0.466851	0.474576	0.474576	0.474576	0.474576
37x53	0.606422	0.606422	0.606422	0.606422	0.606422

Из каждого инстанса выбиралось 5 лучших решений. Таким образом, после чистки от повторений, имеем список из 22 решений. Прогоняя через все остальные инстансы и высчитывая среднюю целевую функцию каждого решения, получаем следующую таблицу.

	binary string	int parametrs	20x20	24x40	30x50	30x90	37x53	average
0	0101110000000	[3, 8, 0, 0, 1, 1]	0.401639	0.415663	0.492228	0.441417	0.599638	0.470117
1	0001111011111	[1, 8, 1, 0, 8, 4]	0.422535	0.448276	0.502618	0.468927	0.594495	0.487370
2	0010011100011	[2, 2, 1, 1, 1, 4]	0.408451	0.436620	0.494681	0.465969	0.591654	0.479475
3	0010011100111	[2, 2, 1, 1, 2, 4]	0.389706	0.443662	0.483516	0.448753	0.594037	0.471935
4	0010011111011	[2, 2, 1, 1, 7, 4]	0.414966	0.455128	0.507937	0.473973	0.595256	0.489452
5	1000010110011	[5, 2, 0, 1, 5, 4]	0.401361	0.454545	0.478873	0.445040	0.586924	0.473349
6	1000110001111	[5, 4, 0, 0, 4, 4]	0.405229	0.442953	0.478947	0.451872	0.605920	0.476984
7	1000111111001	[5, 4, 1, 1, 7, 2]	0.419580	0.437500	0.482051	0.454308	0.596313	0.477951
8	1001101011010	[5, 7, 1, 0, 7, 3]	0.416058	0.458599	0.486339	0.460000	0.597976	0.483794
9	1001111111101	[5, 8, 1, 1, 8, 2]	0.427632	0.454545	0.505495	0.475000	0.605436	0.493621
10	1011001111100	[6, 5, 1, 1, 8, 1]	0.424460	0.457746	0.500000	0.462396	0.604070	0.489735
11	1011100101101	[6, 7, 0, 1, 4, 2]	0.418919	0.447552	0.497512	0.449036	0.606422	0.483888
12	1011101011111	[6, 7, 1, 0, 8, 4]	0.414966	0.437500	0.492063	0.454039	0.605239	0.480761
13	1011101111110	[6, 7, 1, 1, 8, 3]	0.416667	0.461538	0.500000	0.456000	0.601843	0.487210
14	1011111110111	[6, 8, 1, 1, 6, 4]	0.423611	0.458904	0.495050	0.464865	0.595256	0.487537
15	1101010110110	[7, 6, 0, 1, 6, 3]	0.424658	0.449367	0.502591	0.442359	0.606422	0.485079
16	1101110011011	[7, 8, 0, 0, 7, 4]	0.407692	0.454545	0.500000	0.445682	0.602241	0.482032
17	1110111100100	[8, 4, 1, 1, 2, 1]	0.415584	0.435583	0.444444	0.431579	0.602727	0.465984
18	1110111101100	[8, 4, 1, 1, 4, 1]	0.420690	0.449664	0.494444	0.445378	0.602241	0.482484
19	1111000111111	[8, 5, 0, 1, 8, 4]	0.423611	0.462025	0.505319	0.423267	0.605920	0.484029
20	1111001101001	[8, 5, 1, 1, 3, 2]	0.418750	0.452830	0.467337	0.459610	0.595256	0.478757
21	1111010101001	[8, 6, 0, 1, 3, 2]	0.424658	0.443662	0.479798	0.434109	0.600746	0.476594
22	1111110101001	[8, 8, 0, 1, 3, 2]	0.418919	0.456954	0.487179	0.440217	0.605920	0.481838

Нулевое решение — это набор параметров, с которым изначально запускался алгоритм в прошлой лабораторной. Среднее значение целевой функции 0.47. Как видно, с помощью подбора параметров удалось повысить это значение.
А лучший результат достигается в 9-м решении (0.493621, в каждом из инстансов был достигнут оптимум) при значении параметров 5,8,1,1,8,2.

Первый параметр отвечает за то, по сколько элементов будет в начальных кластерах (от 1 до 8). Второй — сколько раз надо запустить алгоритм целиком, чтобы получить лучшее решение (от 1 до 8). Третий — в каком порядке запускаются функции в local search (поменять строчку/поменять столбец или наоборот). Четвертый — в каком порядке запускаются функции в shaking (merge/split или наоборот). Пятый — сколько раз повторяется shaking на одной операции, если нет улучшений (от 1 до 8). Шестой — сколько случайных попыток разбиения или слияния будет сделано для достижения лучшего результата (от 1 до 4).

По итогу получается, что чтобы достичь лучшего результата, нужно запустить алгоритм, где изначально в кластерах будет по 5 элементов;
алгоритм повторится 8 раз;
в ls сначала будет меняться столбец, а потом строчка;
в shaking будет сначала split, потом merge;
shaking на одной окрестности будет повторяться 8 раз, пока нет улучшений;
а при разбиении или слиянии будет сделано две случайные попытки и выбран лучший результат.

Если говорить о структуре проекта, то Genetic_algoritm.ipynb — файл с реализацией генетики, VNS_ILS_for_GA.py — модуль с измененными функциями из предыдущей лабораторной, чтобы алгоритм работал с параметрами, а VNS_ILS_for_GA.ipynb — то же самое, но в читабельном формате.