Seminarski rad iz predmeta

Računarska inteligencija

Najbliži string

(eng. Closest String)

David Gavrilović 294/2015

Marko Veljković 43/2015

Dr Aleksandar Kartelj

Dr Stefan Mišković

Sadržaj

Uvodni deo

Opis problema

Problem najbližeg stringa se može definisati na sledeći način:

Dat je konačan skup stringova **S** = {s_1, s_2, …, s_N} formiranih nad konačnom azbukom **A** = {c_1, …, c_k}, od kojih je svaki dužine M. Treba pronaći string **t** koji predstavlja medijanu skupa **S**, dužine M koji minimizuje **d**, takav da je za svaki string $\mathbf{s_i}$ koji pripada **S**, važi $\mathbf{dist_H(t, s_i) \leq d}$. Rastojanje $\mathbf{dist_H(t, s_i)}$ predstavlja Hamingovo rastojanje između stringova **t** i $\mathbf{s_i}$. Hamingovo rastojanje stringova a i b jednake dužine, predstavlja broj različitih karaktera na istim pozicijama u stringovima. Ovaj problem spada u NP-teške probleme.

Iako postoje dobre aproksimacije algoritma za rešavanje ovog problema, kao i egzaktni algoritmi za fiksirano **d**, ne postoje pokušaji da se problem reši egzaktno za opšti slučaj.

Primer problema:

Neka je S = {‘ATTT’, ‘AGGG’, ‘ACCC’}. Jedno od rešenja je ‘ACGT’ zbog toga što je rastojanje od tog stringa do svih ostalih jednako, kao i takvo da je najmanje moguće najveće rešenje.

Primena

Rešenja ovog problema pronalaze primenu u poljima kao što su biološka izračunavanja (eng. *Computational Biology*), molekularna biologija (eng. *Molecular biology*), na primer za dizajniranje novih genetskih lekova koji imaju strukturu sličnu datom skupu postojećih sekvenci RNK, teoriji kodiranja (eng. *Coding theory*), na primer u određivanju najboljeg načina enkriptovanja skupa poruka.

Prethodni radovi

- Razvoj primene bioloških izračunavanja zahtevao je proučavanje optimizacionih problema nad sekvencama karaktera. Neka početna istraživanja su koristila statističke metode [1]
- Ming Li je prvi otkrio kombinatornu prirodu ovog problema i dokazao da je NP-težak. Takođe je opisao neke primene u molekularnoj biologiji [2]
- U [3] Gram je pokazao da se za fiksiranu vrednost **d** ovaj problem može rešiti u polinomijalnom vremenu.

Opis rešenja

Opis klase

Napravili smo klasu ‘ClosestString’ koja rešava problem korišćenjem genetskog algoritma. Atributi su parametri opšteg genetskog algoritma, poput veličine populacije, broja iteracija, verovatnoće mutacije, veličina turnira i drugih.

```
// members
std::vector<std::string> _setOfStrings;
std::vector<char> _allowedGeneValues;
size_t _length;
unsigned _numOfIterations;
unsigned _generationSize;
double _mutationRate;
unsigned _reproductionSize;
unsigned _tournamentK;
double _crossoverProb;
Chromosome _best;
```

Klasa sadrži metode za rad sa genetskim algoritmom.

```
public:
   //First, temporary constructor, without parameters
   ClosestString(const std::vector<std::string> &setOfStrings, const std::vector<char> &allowedGeneValues);

   //Main constructor, with all parameters for genetic algorithm
   ClosestString(const std::vector<std::string> &setOfStrings, const std::vector<char> &allowedGeneValues,
              unsigned numOfIterations, unsigned generationSize, double mutationRate,
              unsigned tournamentK, double crossoverProb);

   //Main function, all begin here
   void optimize();

private:
   //Create random population, from allowed gene values
   std::vector<Chromosome> initialPopulation();
   //tournament selection, with parametar K, which goes from GenerationSize/5 to GenerationSize
   std::vector<Chromosome> selection(const std::vector<Chromosome> &population);
   //Use selected chromosomes, do uniform crossover on 2 parents to get 2 children,
   //some of children are mutated and add to vector
   std::vector<Chromosome> createGeneration(const std::vector<Chromosome> &forReproduction);
   //Uniform crossover
   std::pair<Chromosome, Chromosome> crossover(const Chromosome &parent1, const Chromosome &parent2);
   //Mutation on random bit of chromosome value, with probability MutationRate
   void mutation(Chromosome &chromo);
   //From all chromosomes that are forwarded as parameter, pick and return the one with best fitness
   Chromosome pickOneTournament(const std::vector<Chromosome> & pop);
   //Calculate fitness for the forwarded string
   int fitness(const std::string & current);
   //Stop conditions for whole program
   bool stopConditions(size_t i, const std::vector<Chromosome> &chromosomes);
```

Opšti rad algoritma

Komponente genetskog algoritma

Karakteristika	Implementacija
Reprezentacija	String (niska karaktera)
Ukrštanje	Ravnomerno ukrštanje
Mutacija	Zamena nasumičnog karaktera nasumičnim karakterom
Selekcija roditelja	Fitnes-srazmerna (turnirska)
Selekcija preživelih	Smena generacija

Opšti genetski algoritam

```
void ClosestString::optimize() {
    std::vector<Chromosome> chromosomes = initialPopulation();

    size_t currIteration = 0;
    while (stopConditions(currIteration, chromosomes)) {
        std::vector<Chromosome> forReproduction = selection(chromosomes);
        chromosomes = createGeneration(forReproduction);

        // Chromosome with smallest fit in population
        _best = *(std::min_element(std::cbegin(chromosomes), std::cend(chromosomes), compare));
        currIteration++;
    }
}

bool compare(Chromosome c1, Chromosome c2){
    return c1.fit < c2.fit;
}
```

Inicijalizacija početne populacije

```
std::vector<Chromosome> ClosestString::initialPopulation(){

    srand(unsigned(time(nullptr)));

    std::vector<Chromosome> initPopulation;
    for (unsigned i = 0; i < _generationSize; ++i){
        std::vector<char> geneticCode;
        //make one string as vector<char> from allowed charachers
        for (size_t j = 0; j < _length; j++){
            size_t pos = unsigned(rand()) % _allowedGeneValues.size();
            geneticCode.push_back(_allowedGeneValues[pos]);
        }
        //make string from vector<char>
        std::string current = std::string(std::cbegin(geneticCode), std::cend(geneticCode));
        initPopulation.push_back(Chromosome{current, fitness(current)});
    }
    return initPopulation;
}
```

Selekcija roditelja

```
std::vector<Chromosome> ClosestString::selection(const std::vector<Chromosome> &population) {
    std::vector<Chromosome> forReproduction;

    for(size_t i = 0, n = population.size(); i < n; ++i){
        forReproduction.push_back(pickOneTournament(population));
    }
    return forReproduction;
}

Chromosome ClosestString::pickOneTournament(const std::vector<Chromosome> &pop) {
    Chromosome bestC{"", INT_MAX};
    for(size_t i = 0; i < _tournamentK; ++i){
        size_t pos = unsigned(rand()) % pop.size();
        if(pop[pos].fit < bestC.fit){
            bestC = pop[pos];
        }
    }
    return bestC;
}
```

Ukrštanje

```
std::pair<Chromosome, Chromosome> ClosestString::crossover(const Chromosome &parent1,
                                                           const Chromosome &parent2) {
    size_t n = parent1.value.size();
    std::string child1, child2;
    child1.resize(n);
    child2.resize(n);

    for (size_t i = 0; i < n; ++i) {
        double tmp = double((double(rand()) / RAND_MAX));
        if (tmp < _crossoverProb) {
            child1.at(i) = parent1.value.at(i);
            child2.at(i) = parent2.value.at(i);
        } else {
            child2.at(i) = parent1.value.at(i);
            child1.at(i) = parent2.value.at(i);
        }
    }
    return {Chromosome{child1, fitness(child1)}, Chromosome{child2, fitness(child2)}};
}
```

Mutacija

```
void ClosestString::mutation(Chromosome &chromo) {

    double mutationImpossible = (double(rand()) / RAND_MAX);
    if (mutationImpossible < _mutationRate) {
        //select position in string that will be changed
        size_t index = unsigned(rand()) % chromo.value.size();
        //select which charachter will be set on selected position in chromosome value
        size_t pos = unsigned(rand()) % _allowedGeneValues.size();

        chromo.value.at(index) = _allowedGeneValues.at(pos);
        chromo.fit = fitness(chromo.value);
    }
}
```

Generisanje nove generacije

```
std::vector<Chromosome> ClosestString::createGeneration(const std::vector<Chromosome> &forReproduction) {

    std::vector<Chromosome> newGeneration;
    for (size_t i = 0; i < _generationSize; i+=2) {
        size_t index1 = unsigned(rand()) % forReproduction.size();
        size_t index2 = unsigned(rand()) % forReproduction.size();

        std::pair<Chromosome, Chromosome> children =
                crossover(forReproduction[index1], forReproduction[index2]);

        mutation(children.first);
        mutation(children.second);

        newGeneration.push_back(children.first);
        newGeneration.push_back(children.second);
    }

    return newGeneration;
}
```

Funkcija prilagođenosti

```
int hamingDistance(const std::string & s1, const std::string & s2){
    return std::inner_product(std::cbegin(s1), std::cend(s1), std::cbegin(s2), 0,
                              [] (int left, int right) {return left + right;},
                              [] (char c1, char c2) {return c1 != c2 ? 1 : 0;});
}

int ClosestString::fitness(const std::string &current) {
    //Getting position in setOfStrings vector,
    //that belongs to string which is furthest from Current chromosome value from population
    auto i = std::max_element(std::cbegin(_setOfStrings), std::cend(_setOfStrings),
                             [current](std::string s1, std::string s2){
                                     return hamingDistance(s1, current) < hamingDistance(s2, current);});
    //Calculate their distance
    return hamingDistance(*i, current);
}
```

Kriterijumi zaustavljanja

```
bool ClosestString::stopConditions(size_t currIteration, const std::vector<Chromosome> &) {
    if (currIteration >= _numOfIterations)
        return false;

    if (_best.fit == 0)
        return false;

     return true;
}
```

Rezultati

Upoređivanje sa drugim rešenjima iz literature

Upoređivanje 1

Testovi rešenja koji su dobijeni u [6]. Dobijeni su primenom B&B algoritma [7], kao i B&B algoritma poboljšanog heuristikom [8]. Dati rezultati su izvršeni na sistemu sa specifikacijama: Pentium 4 CPU, 2.8 GHz i 512MB RAM na WindowsXP.

instance			B&B				heuristic		
num	n	m	sol	LB	#nodes	time(s)	sol	time(s)	ratio
1	10	300	174	174	5652	10.87	181	5	1.04
2	10	400	235	235	647	2.92	239	7	1.02
3	10	500	291	290	1	0.93	301	10	1.03
4	10	600	348	348	1225	8.97	354	11	1.02
5	10	700	404	404	933	9.46	411	13	1.02
6	10	800	459	458	2579	17.52	472	15	1.03
7	15	300	186	185	1701997	4024.78	191	8	1.03
8	15	400	247	247	3025	10.17	256	12	1.05
9	15	500	303	301	201845	780.41	315	15	1.04
10	15	600	369	367	4994	22.83	375	18	1.02
11	15	700	433	427	704590	3925.63	437	20	1.01
12	15	800	492	489	616061	3877.36	500	23	1.02
13	20	300	190	190	108281	358.90	196	12	1.03
14	20	400	255	253	345031	1261.30	263	17	1.03
15	20	500	316	316	166950	671.74	327	22	1.03
16	20	600	379	378	632070	3868.59	391	24	1.03
17	20	700	442	442	3755	22.88	449	28	1.02
18	20	800	506	505	512329	3802.08	518	33	1.02
19	25	300	196	195	1405903	4004.59	204	16	1.04
20	25	400	260	259	843502	3874.79	266	21	1.02
21	25	500	322	321	658143	3812.68	334	27	1.04
22	25	600	390	389	577236	3842.66	398	30	1.02
23	25	700	454	453	501478	3786.70	462	36	1.02
24	25	800	516	516	97408	698.46	532	41	1.03
25	30	300	198	197	1222037	3975.23	203	18	1.03
26	30	400	265	264	831860	3875.11	271	26	1.02
27	30	500	328	327	589934	3845.15	336	32	1.02
28	30	600	394	393	456213	3765.45	405	37	1.03
29	30	700	459	458	406539	3824.78	468	43	1.02
30	30	800	525	524	367610	3752.77	542	49	1.03

Slika 1: $\mathbf{A} = \{A, C, T, G\}$

Rezultati dobijeni primenom našeg rešenja

N	M	NumberOfIterations	PopulationSize	MutationRate	TournamentSize	CrossoverProbability	LastIteration	BestFit	Elapsed time(ms)
10	500	100	100	0.02	20	0.5	100	357	2079.932451
10	500	250	100	0.02	20	0.5	250	357	5163.505316
10	500	500	100	0.02	20	0.5	500	366	10242.137432
10	800	100	100	0.02	20	0.5	100	572	3318.776131
10	800	250	100	0.02	20	0.5	250	575	8226.156235
10	800	500	100	0.02	20	0.5	500	570	16776.466608
20	500	100	100	0.02	20	0.5	100	371	4097.535372
20	500	250	100	0.02	20	0.5	250	366	10258.228302
20	500	500	100	0.02	20	0.5	500	365	20455.005169
20	800	100	100	0.02	20	0.5	100	580	6811.946630
20	800	250	100	0.02	20	0.5	250	582	16503.336668
20	800	500	100	0.02	20	0.5	500	582	32749.498606
30	500	100	100	0.02	20	0.5	100	376	6229.389191
30	500	250	100	0.02	20	0.5	250	373	15351.097107
30	500	500	100	0.02	20	0.5	500	371	31072.959661
30	800	100	100	0.02	20	0.5	100	595	9887.326956
30	800	250	100	0.02	20	0.5	250	602	24453.977108
30	800	500	100	0.02	20	0.5	500	601	48187.983036

Slika 2: ***A*** *= {A, C, T, G}*

Upoređivanje 2

Rezultati rešenja iz [6] dobijeni primenom istog algoritma i heuristike kao u prvom upoređivanju. Specifikacije sistema su takođe iste.

	instance		B&B				heuristic		
num	n	m	sol	LB	#nodes	time(s)	sol	time(s)	ratio
31	10	400	156	155	2687979	4141.82	171	5	1.10
32	10	500	190	189	2184136	4189.93	190	5	0.00
33	10	600	229	228	1733818	4485.28	247	7	1.08
34	10	800	305	303	1482246	4006.00	306	9	1.003
35	15	300	122	121	2806838	4128.27	130	5	1.07
36	15	400	159	158	2166362	4156.88	161	7	1.01
37	15	500	202	200	1772538	4071.95	213	9	1.05
38	15	600	240	239	1503438	3967.02	246	11	1.03
39	15	700	278	277	1318088	4101.48	323	14	1.16
40	15	800	325	324	1092610	3911.65	333	15	1.02
41	20	300	126	125	2386388	4063.52	133	8	1.06
42	20	400	165	164	1872385	4063.76	191	11	1.16
43	20	500	207	206	1454067	3963.46	218	14	1.05
44	20	600	247	246	1203460	3918.93	265	16	1.08
45	20	700	291	290	1080569	3913.32	311	19	1.07
46	20	800	331	330	944268	3870.64	359	21	1.08
47	25	300	131	130	1943166	4002.80	145	10	1.11
49	25	400	173	172	1541728	3975.91	182	14	1.05
50	25	500	211	211	17158	56.25	228	18	1.08
51	25	600	255	254	1041537	3881.25	271	19	1.06
52	25	700	300	299	893404	3948.62	314	23	1.05
53	25	800	337	336	786798	3903.74	351	26	1.04
54	30	300	133	132	1674260	4026.07	150	13	1.13
55	30	400	174	172	1367616	4002.25	184	16	1.06
56	30	500	217	216	1073655	4029.87	242	22	1.12
57	30	600	263	262	874222	4025.71	296	26	1.13
58	30	700	301	299	773934	4015.73	316	27	1.05

Slika 3: ***A*** *= {0, 1}*

Rezultati dobijeni primenom našeg rešenja

N	M	NumberOfIterations	PopulationSize	MutationRate	TournamentSize	CrossoverProbability	LastIteration	BestFit	Elapsed time(ms)
10	500	100	100	0.02	20	0.5	100	221	4239.798307
10	500	250	100	0.02	20	0.5	250	229	10164.439440
10	500	500	100	0.02	20	0.5	500	229	20338.255644
10	800	100	100	0.02	20	0.5	100	378	6884.086132
10	800	250	100	0.02	20	0.5	250	374	16658.798695
10	800	500	100	0.02	20	0.5	500	373	34675.085306
20	500	100	100	0.02	20	0.5	100	236	8710.000753
20	500	250	100	0.02	20	0.5	250	248	21464.530230
20	500	500	100	0.02	20	0.5	500	238	43263.711929
20	800	100	100	0.02	20	0.5	100	384	14031.580448
20	800	250	100	0.02	20	0.5	250	387	33835.106611
20	800	500	100	0.02	20	0.5	500	387	67745.550156
30	500	100	100	0.02	20	0.5	100	252	12971.709013
30	500	250	100	0.02	20	0.5	250	250	31963.227987
30	500	500	100	0.02	20	0.5	500	246	65096.003532
30	800	100	100	0.02	20	0.5	100	390	20705.399513
30	800	250	100	0.02	20	0.5	250	388	52484.143734
30	800	500	100	0.02	20	0.5	500	396	103839.802980

Slika 4: ***A*** *= {0, 1}*

Upoređivanje 3

Testovi rešenja koji su dobijeni u [9]. Dobijeni su primenom ANT algoritma [10], heuristikom [11], paralelizovana verzija heuristike [12], 'root' heuristika objašnjena u [13] i B&B [14].

Specifikacija sistema:

- ANT: CPU 1.8 GHz 1GB RAM
- H: CPU 2.8 GHz 512MB RAM
- HP: CPU Intel Xeon dual 2 x 1GB RAM

		time (sec)					distance d							
k	n	ANT	H	HP	RT	BB	ANT	H	HP	RT		BB		%BB
10	500	2.11	1.00	-	**0.08**	0.08	317	295	-	**290.8**		290.8		100%
10	1000	7.85	-	1.2	**0.17**	0.17	652	-	590.7	**579.8**		579.8		100%
10	2000	-	-	4.8	**0.45**	0.45	-	-	1178.3	**1161.5**		1161.5		100%
10	5000	-	-	18.9	**1.43**	1.43	-	-	2931	**2901.3**		2901.3		100%
20	500	3.51	2.00	-	**0.33**	2.42	341	325.5	-	**316.6**		316.6		100%
20	1000	11.80	-	3.3	**0.77**	366.79	695	-	651.0	**632.9**	(633.13)	632.5	(632.63)	80%
20	2000	-	-	10.6	**1.08**	2.42	-	-	1295.3	**1262.6**		1262.4		100%
20	5000	-	-	45.0	**5.80**	595.68	-	-	3233.3	**3156.9**	(3155.57)	3156.6	(3155.14)	70%
30	500	6.76	3.00	-	**0.60**	1265.08	351	339.25	-	**328.5**	(328.33)	328.3	(327.67)	30%
30	1000	10.70	-	7.1	**1.32**	950.90	713	-	675.3	**655.5**	(655.6)	655.0	(654.6)	50%
30	2000	-	-	17.9	**3.05**	1262.41	-	-	1347.0	**1307.6**	(1307.67)	1307.5	(1307.33)	30%
30	5000	-	-	72.8	**11.90**	835.64	-	-	3364.0	**3267.7**	(3267.5)	3267.4	(3267)	60%

*Slika 5: |**A**| = 4*

Rezultati dobijeni primenom našeg rešenja

N	M	NumberOfIterations	PopulationSize	MutationRate	TournamentSize	CrossoverProbability	LastIteration	BestFit	Elapsed time(ms)
10	500	100	100	0.02	20	0.5	100	357	2106.479406
10	500	250	100	0.02	20	0.5	250	348	5230.440855
10	500	500	100	0.02	20	0.5	500	350	10396.999359
10	1000	100	100	0.02	20	0.5	100	717	4287.664413
10	1000	250	100	0.02	20	0.5	250	719	10323.371410
10	1000	500	100	0.02	20	0.5	500	714	20882.388115
10	2000	100	100	0.02	20	0.5	100	717	4287.664413
10	2000	250	100	0.02	20	0.5	250	719	10323.371410
10	2000	500	100	0.02	20	0.5	500	714	20882.388115
20	500	100	100	0.02	20	0.5	100	368	4092.712879
20	500	250	100	0.02	20	0.5	250	362	10443.640232
20	500	500	100	0.02	20	0.5	500	368	20387.578964
20	1000	100	100	0.02	20	0.5	100	733	8134.496450
20	1000	250	100	0.02	20	0.5	250	738	20100.555897
20	1000	500	100	0.02	20	0.5	500	736	40335.358858
20	2000	100	100	0.02	20	0.5	100	1486	16516.464233
20	2000	250	100	0.02	20	0.5	250	1496	40327.488661
20	2000	500	100	0.02	20	0.5	500	1483	80606.364489
30	500	100	100	0.02	20	0.5	100	372	6113.292933
30	500	250	100	0.02	20	0.5	250	374	15164.133549
30	500	500	100	0.02	20	0.5	500	373	30629.849195
30	1000	100	100	0.02	20	0.5	100	747	12293.381214
30	1000	250	100	0.02	20	0.5	250	751	30293.049097
30	1000	500	100	0.02	20	0.5	500	743	60312.296152
30	2000	100	100	0.02	20	0.5	100	1488	24464.812994
30	2000	250	100	0.02	20	0.5	250	1484	60608.834028
30	2000	500	100	0.02	20	0.5	500	1486	120390.976906
30	5000	100	100	0.02	20	0.5	100	3734	60367.697001
30	5000	250	100	0.02	20	0.5	250	3740	150128.245354
30	5000	500	100	0.02	20	0.5	500	3737	302905.275822

Slika 6: A = {A, C, T, G}

Rezultati našeg rešenja

NumberOfIterations	PopulationSize	MutationRate	TournamentSize	CrossoverProbability	BestValue	BestFit	Elapsed time(ms)
100	50	0.02	10	0.5	TCGAA	4	36.696672
100	50	0.02	20	0.5	GCTGG	5	36.269903
100	50	0.02	40	0.5	CGTGT	5	37.979126
100	100	0.02	10	0.5	TCGAA	4	69.952965
100	100	0.02	20	0.5	AGATC	5	68.871260
100	100	0.02	40	0.5	CTATA	5	72.498560
100	200	0.02	10	0.5	TCGAA	4	129.417181
100	200	0.02	20	0.5	TCGAA	4	135.240793
100	200	0.02	40	0.5	TCGAA	4	143.632412
500	50	0.02	10	0.5	CTTTA	5	159.780025
500	50	0.02	20	0.5	AGCTA	5	166.418076
500	50	0.02	40	0.5	TCGAA	4	181.087732
500	100	0.02	10	0.5	TCGAA	4	336.433887
500	100	0.02	20	0.5	TTCCA	5	329.453707
500	100	0.02	40	0.5	TCGAA	4	364.855528
500	200	0.02	10	0.5	TCGAA	4	688.050270
500	200	0.02	20	0.5	TCGAA	4	661.804438
500	200	0.02	40	0.5	TCGAA	4	695.769072
1000	50	0.02	10	0.5	CTAGC	5	333.570480
1000	50	0.02	20	0.5	TCACC	5	338.683128
1000	50	0.02	40	0.5	CCATC	5	356.868267
1000	100	0.02	10	0.5	TCGAA	4	640.821457
1000	100	0.02	20	0.5	TCGAA	4	664.465666
1000	100	0.02	40	0.5	GGGTC	5	692.093134
1000	200	0.02	10	0.5	TCGAA	4	1279.572964
1000	200	0.02	20	0.5	TCGAA	4	1306.427240
1000	200	0.02	40	0.5	TCGAA	4	1387.833118

Slika 7: $\mathbf{A} = \{A, C, T, G\}$, $N = 20$, $M = 5$

NumberOfIterations	PopulationSize	MutationRate	TournamentSize	CrossoverProbability	BestValue	BestFit	Elapsed time(ms)
100	50	0.02	10	0.5	GCGCG	4	19.378424
100	50	0.02	20	0.5	GAGCG	4	29.422998
100	50	0.02	40	0.5	CCGCG	4	27.624846
100	100	0.02	10	0.5	GGGCG	4	43.681145
100	100	0.02	20	0.5	CGGCG	4	42.286873
100	100	0.02	40	0.5	GTGCG	4	48.881054
100	200	0.02	10	0.5	TGGCG	4	74.081421
100	200	0.02	20	0.5	CGGCG	4	79.082012
100	200	0.02	40	0.5	CTGCG	4	90.302467
500	50	0.02	10	0.5	TTGCG	4	94.363451
500	50	0.02	20	0.5	GCGCG	4	101.059198
500	50	0.02	40	0.5	CCGCG	4	110.152006
500	100	0.02	10	0.5	GCGCG	4	178.055286
500	100	0.02	20	0.5	CCATG	4	187.984943
500	100	0.02	40	0.5	CCATT	4	209.007263
500	200	0.02	10	0.5	TCGCG	4	349.377871
500	200	0.02	20	0.5	CCATA	4	366.511345
500	200	0.02	40	0.5	CCATT	4	411.045313
1000	50	0.02	10	0.5	TTAGC	4	180.601597
1000	50	0.02	20	0.5	TTAGT	4	192.665339
1000	50	0.02	40	0.5	TTAGG	4	214.732647
1000	100	0.02	10	0.5	TTGCG	4	348.449230
1000	100	0.02	20	0.5	GAGCG	4	373.154402
1000	100	0.02	40	0.5	GAGCG	4	416.526079
1000	200	0.02	10	0.5	CCATT	4	686.534405
1000	200	0.02	20	0.5	CCATC	4	728.376627
1000	200	0.02	40	0.5	CCGCG	4	812.622070

Slika 8: $A = \{A, C, T, G\}$, $N = 10$, $M = 5$

Specifikacije sistema

CPU:	Intel© Core™ i5-7200U CPU @ 2.50GHz × 2
RAM:	7.7 GiB
Grafička kartica:	NVIDIA GeForce 940MX
Operativni sistem:	Linux Mint 19 Cinnamon, 3.8.9
Kompajler:	g++ (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0

Zaključak

U ovom radu, razmatrali smo rešavanje problema najbližeg stringa pomoću heuristike genetskog algoritma. Promenama parametara algoritma, pokušali smo da dođemo do što boljeg rešenja, ali ono na kraju nije bilo dobro kao rešenja iz literature sa kojima smo upoređivali rezultate.

Jedna od mogućnosti unapređivanja algoritma je paralelizacija rada programa, koju planiramo da uradimo u bliskoj budućnosti.

Kao što smo videli, rešenja problema najbližeg stringa, imaju značajnu primenu u oblastima bioinformatike i kriptografije, tako da je preciznost izvršavanja algoritma od presudnog značaja za njegov kvalitet, ali odmah pored preciznosti se može uvrstiti i brzina izračunavanja.

Literatura

[1] - G. Hertz and G. Stormo. Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps. In Lim and Cantor, editors, Proc. 3rd Int'l Conf. Bioinformatics and Genome Research, pages 201   216. World Scientific, 1995.

[2] - M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. Proceedings of the Thirty First Annual ACM Symposium on Theory of Computing, Atlanta, pages 473   482, 1999.

[3] - J. Gramm, R. Niedermeier, and P. Rossmanith. Exact solutions for closest string and related problems. In In Proceedings of the 12th Annual International Symposium on Algorithms and Computation (ISAAC 2001), volume 2223 of Lecture Notes in Computer Science, pages 441   452. Springer Verlag, 2001.

[4] - http://www.math.wsu.edu/faculty/bkrishna/FilesMath574/Papers/IP_ClosestString.pdf

[5] - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.5849&rep=rep1&type=pdf

[6] - http://www.math.wsu.edu/faculty/bkrishna/FilesMath574/Papers/IP_ClosestString.pdf, strane 18-19

[7] - http://www.math.wsu.edu/faculty/bkrishna/FilesMath574/Papers/IP_ClosestString.pdf, strana 12

[8] - http://www.math.wsu.edu/faculty/bkrishna/FilesMath574/Papers/IP_ClosestString.pdf, strana 13

[9] - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.5849&rep=rep1&type=pdf, strana 6

[10] - S. Faro and E. Pappalardo. Ant-CSP: An ant colony optimization algorithm for the closest string problem. In Proc. SOFSEM 2010, volume 5901 of LNCS, pages 370   381. Springer, 2010

[11] - C. Meneses, Z. Lu, C. Oliveira, and P. Pardalos. Optimal solutions for the closest-string problem via integer programming. INFORMS Journal on Computing, 16(4):419   429, 2004

[12] - F. Gomes, C. Meneses, P. Pardalos, and G. Viana. A parallel multistart algorithm for the closest string problem. Computers & Operations Research, 35(11):3636   3643, 2008

[13] - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.5849&rep=rep1&type=pdf, stane 4-5

[14] - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.5849&rep=rep1&type=pdf, strana 4