КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.8 DOI:10.34229/2707-451X.20.2.3

Л.Ф. ГУЛЯНИЦЬКИЙ, С.А. ЧОРНОЖУК

ГЕНЕТИЧНИЙ АЛГОРИТМ З ЖАДІБНИМ СТОХАСТИЧНИМ ОПЕРАТОРОМ СХРЕЩУВАННЯ ДЛЯ ПЕРЕДБАЧЕННЯ ТРЕТИННОЇ СТРУКТУРИ БІЛКА

Розроблено новий генетичний алгоритм, особливістю якого є запропонований жадібний стохастичний оператор схрещування. Застосування пропонованого алгоритму досліджується на задачі передбачення третинної структури білка. Наведено результати обчислювального експерименту.

Ключові слова: *третинна структура білка, комбінаторна оптимізація, генетичні алгоритми, оператор схрещування, стохастичність.*

Вступ. Визначення третинної структури білків є важливою та актуальною проблемою біології. Одним із підходів до формального подання структури білка в просторі є HP модель Діла [1], де кожна з амінокислот, що входить у структуру, належить одному з двох типів: H – гідрофобний тип, P – полярний, а сама структура подається як послідовність амінокислот, що розміщена у деякій (просторовій) ґратці. Сформульована Крістіаном Афінсеном термодинамічна гіпотеза [2] припускає, що розміщення ламаної у цій ґратці характеризує вільну енергію структури білка, що дозволяє розглядати задачу мінімізації енергії як мінімізацію певної функції у дискретному просторі за заданих обмеженнях, чим проблема може зводитися до спеціальної задачі комбінаторної оптимізації. При розв'язуванні цієї та інших задач комбінаторної оптимізації застосування знайшли різні метаевристичні методи, такі як алгоритм імітаційного відпалу [3], алгоритм оптимізації мурашиними колоніями [4–6]. Низка робіт присвячена розробці та застосуванню генетичних та міметичних алгоритмів (загальний опис див., наприклад, [7–10]). Варто згадати такі важливі дослідження, як самоадаптивний міметичний алгоритм Красногора та Сміта [11], а також міметичний алгоритм Бацолі та Тетаманці [12] (який одним з перших знайшов значення глобального мінімуму вільної енергії на 8 з 10 відомих послідовностях, вперше пропонованих у [13]), модифікації класичних генетичних алгоритмів [14, 15], генетичні алгоритми з вдосконаленими операторами мутації та схрещування на базі статистичної інформації [16] та новітній ймовірнісний підхід до задачі передбачення третинної структури білків [17]. Важливою частиною генетичних та міметичних алгоритмів, від якої суттєво залежить їх ефективність, є оператор схрещування (crossover). У всіх вищеназваних дослідженнях використовуються класичні кросовери, такі як одноточковий, двоточковий, багатоточковий, які, втім, не аналізують можливі значення цільової функції нащадків (у згаданих дослідженнях

це значення враховується в інших частинах алгоритмів). У статті пропонується новий жадібний оператор схрещування, який враховує можливе значення цільової функції нащадка, що утворюється у результаті застосування цього оператора.

Математична постановка задачі. У роботі використовується HP модель Діла. У такій моделі кожна амінокислота (мономер) $\xi_i, i=\overline{1,n}$, яка входить до первинної структури білка, належить одному з двох типів: H – гідрофобний тип, P – полярний, а сама третинна структура подається як послідовність амінокислот, що розміщена у деякій (просторовій) ґратці у вигляді неперервного ланцюга [18]. У цій роботі використовується кубічна ґратка. Важливо зазначити, що у такого ланцюга не має бути самоперетинів. Тоді цільова функція (енергія утвореної третинної структури) обчислюється за формулою

$$F(x) = - \sum_{1 \le i \le j-2 \le n-2} I(U(\xi_i), U(\xi_j)) h(\xi_i) h(\xi_j), \qquad (1)$$

де $x \in X$ – неперервний ланцюг без самоперетинів довжиною n, X – простір усіх можливих неперервних ланцюгів довжиною n без самоперетинів, розміщених у кубічній ґратці, $U(\xi_i)$ – вузол у кубічній ґратці, який містить i-й мономер ланцюга (заданий декартовими координатами), а

$$I(U_1, U_2) = \begin{cases} 1, & \text{якщо вузли } U_1 \text{ та } U_2 \text{ сусідні в кубічній ґратці,} \\ 0, & \text{в іншому разі,} \end{cases}$$

$$h(\xi) = \begin{cases} 1, & \text{якщо } \xi = \text{H}, \\ 0, & \text{якщо } \xi = \text{P}. \end{cases}$$

Задача полягає у тому щоб знайти такий допустимий ланцюг $x_{opt} \in X$, що

$$x_{opt} = \arg\min_{x \in X} F(x). \qquad (2)$$

Конкретний ланцюг із X зручно подавати не у вигляді абсолютних координат вузлів, у яких він розташований, а за допомогою внутрішнього відносного кодування [19], при якому вважається, що ланцюг починається в точці (0, 0, 0), а надалі кожен новий сегмент ланцюга задається відносним поворотом, щодо свого попереднього положення (вправо на 90 градусів – right, вліво на 90 градусів – left, догори на 90 градусів – up, донизу на 90 градусів – down, рух прямо без поворотів – front). Наприклад, ланцюг ABCDEF (рис. 1) при заданій початковій орієнтації споглядання, як на рисунку, буде мати кодування (front, up, right, down, down).

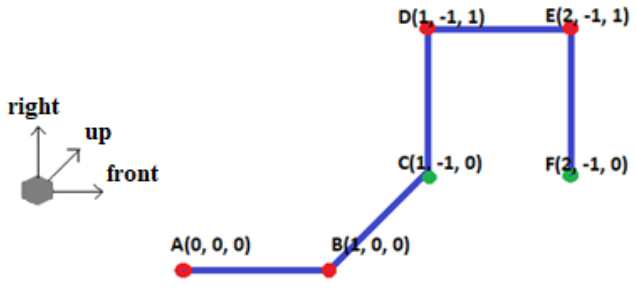

РИС. 1. Ланцюг ABCDEF має відносне кодування (front, up, right, down, down)

У цій роботі використовується відносне кодування, а початкова орієнтація споглядання задається за рахунок двох перших неоднакових поворотів. Перший з них обов'язково буде front, а другий – up. Така початкова орієнтація робить відносне кодування інваріантним відносно будь-якого повороту, кратного 90 градусам навколо будь-якої з координатних осей (рис. 2). Використання такого підходу зменшує простір X, а значить робить пошук x_{opt} у X більш ефективним. Нагадаємо, що кодування $Enc(U(\xi_1),U(\xi_2),...,U(\xi_n))$ ланцюга x, який подається послідовності мономерів $\xi_1,\xi_2,...,\xi_n$, є інваріантним відносно довільного відображення $f:\mathbb{Z}^3\to\mathbb{Z}^3$, якщо ґратка та відношення сусідства в ній інваріантні відносно f, та $Enc(U(\xi_1),U(\xi_2),...,U(\xi_n)) = Enc(f(U(\xi_1)),$ $f(U(\xi_2)),...,f(U(\xi_n)))$ [18].

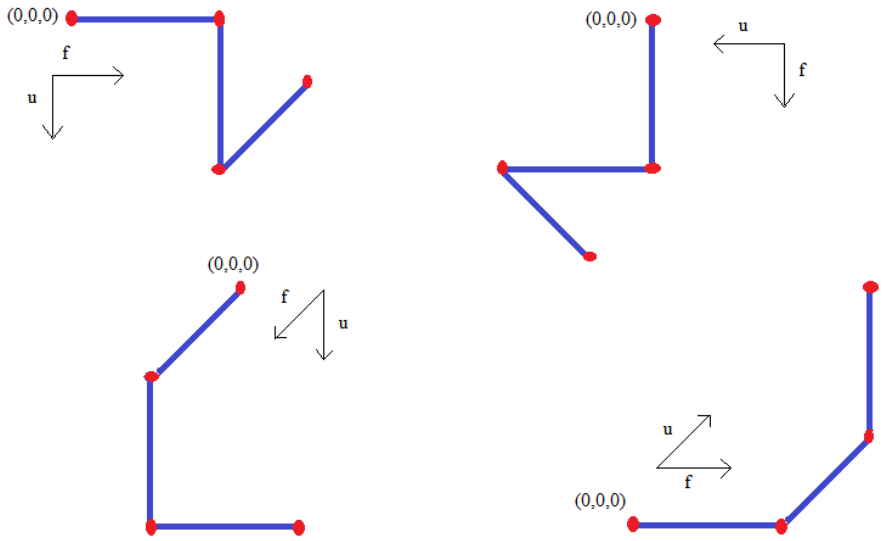

РИС. 2. Три ланцюги, утворені шляхом повороту лівого зверху на 90 градусів навколо кожної з осей. Орієнтуючи споглядання так, як показано, кожне з відносних кодувань буде (front, up, right)

Для розв'язування задачі (2) з цільовою функцією (1) далі пропонується генетичний алгоритм SGCGA (Stochastic greedy crossover genetic algorithm). Його головна особливість – це спеціальний оператор схрещування, який враховує значення цільової функції, що обчислене на згенерованому варіанті потенційного нащадка.

Загальна схема генетичного алгоритму та його параметри. Будемо називати хромосомою будь-яке відносне кодування ланцюга x з X. Тоді геном будемо називати будь-який поворот із цього кодування.

Введемо позначення:

μ – кількість хромосом на кожному поколінні популяції;

η – кількість хромосом із популяції, які використовуються в операторі схрещування;

ε – кількість найкращих хромосом у тимчасовій популяції, що гарантовано потрапляють у нову популяцію (реалізація принципу елітизму), $\varepsilon < \mu$;

σ – частина хромосом, які мутують після елітарного відбору;

I – кількість ітерацій без оновлення рекорду, після яких алгоритм завершується.

Як вже зазначалося, SGCGA відрізняється від класичних реалізацій генетичного алгоритму [7, 10] нестандартним оператором схрещування. Цей оператор схрещування жадібно відбирає най-

кращого нащадка з потенційно можливих з урахуванням цільової функції. Зазвичай значення цільової функції враховується в міметичних алгоритмах при локальному пошуці для утворення мімів [12], але оператор схрещування там використовується стандартний (одноточковий, двоточковий, багатоточковий тощо). Також у підході, описаному в алгоритмі, хромосоми в популяціях з часом видозмінюються за рахунок стохастичності оператора схрещування та оператора мутації. Роботу операторів схрещування та мутації подано далі.

Загальна схема пропонованого алгоритму виглядає так:

procedure *Genetic*();

generation = деяка початкова популяція хромосом розмірністю μ;

x_optimal = довільна хромосома розв'язку з *generation*;

energy_optimal := F(x_optimal);

i = поточна кількість ітерацій без оновлення рекорду*;*

i := 0*;*

while *i* < *I* **do**

generation_temp := *generation;*

generation := $\varnothing$;

crossed = Crossover(generation_temp, η*);*

j:=0*;*

for each *c* **in** *crossed* **do**

if *c* **not in** *generation_temp* **then**

Додати *checked* до *generation_temp;*

endif

endfor

Додати ε хромосом з *generation_temp* з найменшим значенням цільової функції до *generation,* видаливши їх з *generation_temp;*

k := ε;

mut:= round$((\mu-\varepsilon)\sigma)$;

while *k* < ε + *mut* **do**

mutation_temp := виконати мутацію випадково вибраної хромосоми *chromosome_temp* з *generation_temp;*

if *mutation_temp* **not in** *generation_temp* **and** *mutation_temp* **not in** *generation* **then**

chromosome_temp := mutation_temp;

endif

Додати *chromosome_temp* в *generation,* видалити *chromosome_temp* з *generation_temp;*

k:=k+1*;*

endwhile

while *k* < μ **do**

Випадковим чином обрати хромосому з *generation_temp* та додати її до *generation,* видалити її з *generation_temp;*

k:=k+1*;*

endwhile

current_optimal := обрати хромосому *з generation*, яка відповідає найменшому значенню цільової функції*;*

energy := F(current_optimal);

if *energy* < *energy_optimal* **then**

x_optimal := current_optimal;

energy_optimal := energy;
i := 0*;*
else then
i := i + 1*;*
endif
endwhile
return {*x_optimal, energy_optimal*};
end

Опис стратегії відбору та оператора схрещування. Хромосоми для схрещування пропонується обирати шляхом турнірної стратегії, близької до стратегії з [15], а з двох випадковим чином вибраних пар хромосом відповідно обиралися кращі у кожній парі та схрещувалися між собою. Обрані батьки викреслювалися з кандидатів, а процес повторювався, поки не відібрано необхідну кількість нащадків. Таким чином, наприклад, при $\mu = 2000$ після етапу схрещування утвориться 1000 потенційно нових хромосом.

Пропонується новий стохастичний жадібний оператор схрещування.

procedure *Crossover*(*chromosome1, chromosome2*****);**

Утворити *parent1, parent2* шляхом переведення відносного кодування *chromosome1, chromosome2* в абсолютне*;*

while можливо повернути *parent2* навколо деякої осі координат у $\mathbb{Z}^3$ на кут кратний 90 градусам проти годинникової стрілки, та цей поворот ще не був розглянутий **do**
parent_temp:= Повернути *parent2* на обраний кут навколо обраної осі;
j:=0;
while *j < N – 2* **do**
var00 := parent1[0, *j*]*;*
var01 := parent_temp[j + 2, N - 1);
temp_rotation := *parent1*[j + 1];
Випадковим чином замінити *parent1*[j + 1] на будь-який інший поворот (включно з поточним);
var0 := var00 + parent1[j + 1] + *var01;*
parent1[j + 1] *:= temp_rotation;*
if *var0* без самоперетинів **then**
e0 := F(var0);
Додати *(var0, e0)* до *possible_children;*
endif
endwhile
j:=0;
while *j < N – 2* **do**
var10 := parent_temp[0, *j*]*;*
var11 := parent1[j + 2, N - 1);
temp_rotation := *parent_temp*[j + 1];
Випадковим чином замінити *parent_temp*[j + 1] на будь-який інший поворот (включно з поточним);
var1 := var00 + parent_temp[j + 1] + *var01;*
parent1[j + 1] *:= temp_rotation;*
if *var1* без самоперетинів **then**
e1 := F(var1);

Додати *(var1, e1)* до *possible_children;*
endif
endwhile
endwhile

p := Обрати пару з *possible_children* з найменшим значенням цільової функції. Якщо таких пар декілька, то обрати ту, в якій абсолютне кодування породжує ланцюг з меншим середньоквадратичним відхиленням координат вузлів з *H*-мономерами у ґратці;

Утворити *result* шляхом переведення абсолютно кодування *p*[0] назад у відносне;

return result;

Необхідно зазначити, що, оскільки описане раніше відносне кодування інваріанте відносно оператора повороту на кут кратний 90 градусам навколо будь-якої з координатних осей, то для виконання початкового повороту відносне кодування необхідно перевести в абсолютне [19].

Початковий поворот – це важлива компонента оператора, яка дозволяє знайти нащадків, що відповідають потенційно кращим значенням цільової функції. Наприклад, нехай після переведення батьків у абсолютне кодування маємо два ланцюги, що відображаються абсолютними кодуваннями (down, right, down, right) та (front, right, front, right) з послідовністю мономерів HHHHP (рис. 3). Якщо виконувати пропонований оператор схрещування без повороту, то неможливо отримати нащадка зі значенням цільової функції, відмінної від 0, проте якщо спочатку повернути другий ланцюг на 90 градусів за годинниковою стрілкою навколо осі left, отримавши при цьому абсолютне кодування (up, right, up, right), то потенційно можна отримати нащадка (down, right, up, right) з відповідним значенням цільової функції 1 (рис. 4).

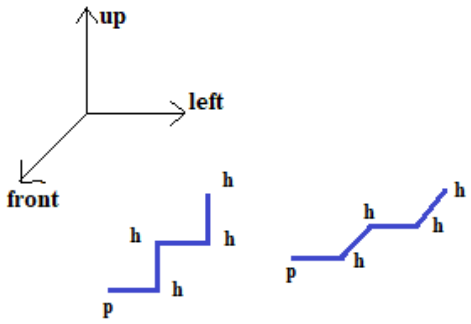

РИС. 3. Ланцюги, які мають абсолютні кодування (down, right, down, right) та (front, right, front, right) відповідно

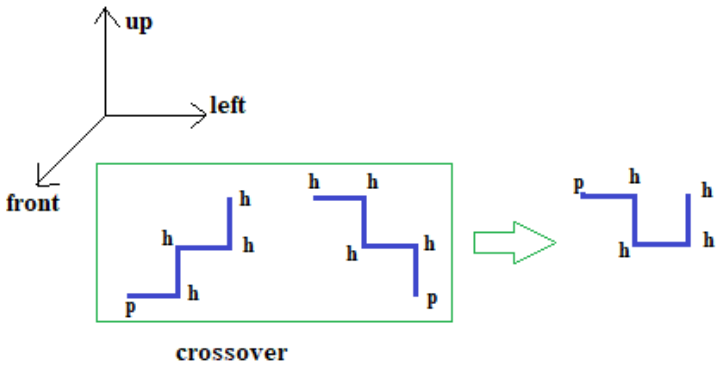

РИС. 4. Приклад виконання оператора схрещування після початкового повороту

Опис оператора мутації. Обчислювальна схема оператора мутації виглядає так:

procedure *Mutation*(*chromosome*);**

Утворити *original* шляхом переведення відносного кодування *chromosome,* в абсолютне.

j := 0*;*

probability_threshold := 0.05*;*

while *j* < *length(chromosome)* **do**

val := *random*[0;1];

if *val* <= *probability_threshold* **then**

Замінити поворот *original*[j] на інший поворот з абсолютним кодуванням у кубічній ґратці, обраний випадковим чином;

endif

j := *j* + *1;*

endwhile

Утворити *result* шляхом переведення абсолютно кодування *original* назад у відносне.

return result;

Повороти при абсолютному кодуванні фактично є базисними векторами в кубічній ґратці [18] , тому їх L2-норма рівна одиниці. Отже, необхідність утворення абсолютного кодування для вхідної хромосоми зумовлюється тим, що заміна певного повороту $\vec{a}$ на поворот $\vec{b}$ при абсолютному кодуванні забезпечує те, що всі інші мономери ланцюга, які закодовані в даному абсолютному кодуванні після повороту $\vec{a}$, змістяться у вузлах кубічної ґратки на зсув $\vec{b}-\vec{a}$ з L2-нормою не більшу за 2 (максимальне значення 2 утворюється при заміні повороту $\vec{a}$ на $-\vec{a}$). Це означає, що мутація не суттєво впливатиме на розміщення ланцюга в кубічній ґратці, що відповідає хромосомі. Зміна повороту у відносному кодуванні, на противагу, може сильно змінити положення відповідного ланцюга у ґратці, що не є бажаним для оператора мутації.

Обчислювальний експеримент. Розроблений алгоритм SGCGA тестувався на 10 реальних значеннях білків довжиною 48, які широко використовуються при аналізі алгоритмів розв'язування задачі, і вперше запропоновані у [13] (див. табл.1). Обчислювальний експеримент проводився на персональному комп'ютері Apple MacBook Pro Touch Bar 13 2019 з процесором 1.4 GHz Quad-Core Intel Core i5 та оперативною пам’яттю 8 GB 2133 MHz LPDDR3.

Параметри алгоритму SGCGA обрані на основі попередніх досліджень так:

$\mu = 10000;\ \eta = 2000;\ \varepsilon = 500;\ \sigma = 0.2;\ I = 150.$

Для кожної з 10 послідовностей зроблено 20 прогонів і підраховувалось найменше ($F_{\min}$) та середнє (F_{avg}) значення цільової функції (енергії). Результати порівнюються з результатами міметичного алгоритму (MA) [12], генетичних алгоритмів (GAHP, AGAHP відповідно) [14, 15] та алгоритму AHEDA [17] , оскільки вони змогли знайти глобальний мінімум для багатьох послідовностей та показали високі результати в середньому.

ТАБЛИЦЯ 1. Номер та код послідовності

Номер послідовності	PH код послідовності
1	HPHHPPHHHHPHHHPPHHPPHPHHHPHPHHPPHHPPPHPPPPPPPPHH
2	HHHHPHHPHHHHHPPHPPHHPPHPPPPPPHPPHPPPHPPHHPPHHHPH

Закінчення таблиці 1

Номер послідовності	PH код послідовності
3	PHPHHPHHHHHHPPHPHPPHPHHPHPHPPPHPPHHPPHHPPHPHPPHP
4	PHPHHPPHPHHHPPHHPHHPPPPHHHHHPPHPHHPHPHPPPPHPPHPHP
5	PPHPPPPHPHHHHPPHHHHPHHPHHHPPHPHPHPPHPPPPPPPHHPHHPH
6	HHHPPPPHHPHPHHPHHPHHPHPPPPPPPHPHPPHPPPPHPPHHHHHHPH
7	PHPPPPHPHHHPHPHHHHPHHPHHPPPHPHPPPHHHPPHHPPHHPPPH
8	PHHPHHHPHHHHPPHHHPPPPPPHPHHPPHHPHPPPHHPHPHPHHPPP
9	PHPHPPPPHPHPHPPHPHHHHHHPPHHHPHPPHPHHPPHPHHHPPPPH
10	PHHPPPPPPHHPPPHHHPHPPHPHHPPHPPHPPHHPPHHHHHHHPPHH

Основні результати наведено в табл. 2, де $F_{\min}$ – рекордно мінімальне значення енергії, F_{avg} – середнє для певної кількості прогонів, F_{opt} – відоме оптимальне значення цільової функції для кожної з послідовностей.

ТАБЛИЦЯ 2. Результати обчислювального експерименту

№	F_{opt}	SGCGA		MA	GAHP		AGAHP		AHEDA	
		$F_{\min}$	F_{avg}	$F_{\min}$	$F_{\min}$	F_{avg}	$F_{\min}$	F_{avg}	$F_{\min}$	F_{avg}
1	–32	–32	–31.7	–32	–32	–30.7	–32	–29.98	–31	–29.5
2	–34	–34	–33.1	–34	–34	–31.2	–34	–31.11	–34	–32.3
3	–34	–34	–33.1	–34	–34	–32	–32	–30.41	–34	–32.4
4	–33	–33	–32.1	–33	–33	–31.1	–33	–30.93	–33	–31.4
5	–32	–32	–31.05	–32	–32	–30.5	–31	–29.81	–31	–29.8
6	–32	–32	–31.0	–32	–32	–29.8	–32	–29.32	–32	–30.1
7	–32	–31	–30.3	–31	–32	–29.8	–32	–28.32	–32	–30.3
8	–31	–31	–30.0	–31	–31	–29.3	–31	–28.26	–30	–28.5
9	–34	–34	–32.65	–33	–34	–31.9	–34	–30.98	–34	–32.3
10	–33	–33	–31.8	–33	–33	–31	–33	–30.06	–31	–29.5

Висновки. На цьому відомому наборі задач SGCGA показав краще середнє значення, ніж усі алгоритми, з якими він порівнювався. До того ж, SGCGA знайшов оптимальне значення цільової функції на 9 із 10 послідовностях, що краще, ніж у MA, AGAHP та AHEDA. У цьому компоненті SGCGA досяг гірших результатів, ніж GAHP (10 із 10), хоча, як зазначалося раніше, показав кращі

середні значення, ніж GAHP. Отримані результати підтверджують перспективність подальших досліджень алгоритма.

Серед перспективних напрямів даних досліджень зазначимо.

1. Для пропонованого оператора схрещування вибір здійснювався на базі класичного одноточкового кросоверу. Перспективним є використання двоточкового кросоверу як базового, що потенційно покращить значення цільової функції нащадка. Також важливо зазначити, що перспективним напрямком досліджень є подання ланцюга у вигляді *q*-кодування кватерніонами [18], що істотно знижує трудомісткість деяких операцій. Це, в свою чергу, знижує трудомісткість оператора схрещування, а значить дозволить застосовувати більш складні та ефективні кросовери [7].

2. Питання вибору параметрів ГА залишається відкритим. Пропонований набір параметрів визначався емпірично, але подальші дослідження щодо формалізації вибору параметрів можуть покращити отримувані алгоритмом результати.

3. Цікавим напрямком дослідження є введення деяких додаткових операцій в алгоритм, таких як pull move [19], що показало свою високу ефективність у покращенні цільової функції для ланцюгів.

Список літератури

1. Dill K.A. Theory for the folding and stability of globular proteins. *Biochemistry*. 1985. **24**. P. 1501–1509.
2. Anfinsen C.B., Haber E., Sela M., White Jr.F.H. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *In Proceedings of the National Academy of Sciences of the USA*, 1961. Vol. **47**. P. 1309–1314.
3. Чорножук С.А. Новий алгоритм імітаційного відпалу для передбачення структури білків. *Компьютерная математика*. 2018. С. 118–124.
4. Dorigo M., Stützle T. Ant colony optimization. Cambridge (MA): MIT Press, 2004.
5. Shmygelska A., Hoos H.H. An ant colony optimization algorithm for the 2D and 3Dhydrophobic polar protein folding problem. *BMC Bioinformatic*s. 2005. **6** (30). P. 30–52.
6. Hulianytskyi L.F., Rudyk V.O. Development and analysis of the parallel ant colony optimization algorithm for solving the protein tertiary structure prediction problem. *Information Theories and* Applications. 2014. **21** (4). P. 392–397.
7. Глибовець М.М., Гулаєва Н.М. Еволюційні алгоритми. К.: НаУ-КМА, 2013.
8. Whitley D. Next Generation Genetic Algorithms: A User's Guide and Tutorial. *Handbook of Metaheuristics. Springer Int. Publ. AG*. 2019. P. 245–274.
9. Moscato P., Cotta C. An Accelerated Introduction to Memetic Algorithms. *Handbook of Metaheuristics. Springer Int. Publ. AG*. 2019. P. 275–309.
10. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. Видавничо-поліграфічний центр "Київський університет", 2016. 142 с.
11. Krasnogor N., Smith J. A memetic algorithm with self-adaptive local search: TSP as a case study. *In GECCO 2000: Proceedings of the Genetic and Evolutionary Computation Conference*. 2000. P. 987–994.
12. Bazzoli A., Tettamanzi A. G. B. A Memetic Algorithm for Protein Structure Prediction in a 3D-Lattice HP Model. *Applications of Evolutionary Computing*. 2004. 3005. P. 1–10.
13. Yue K., Fiebig K.M., Thomas P.D., Chan H.S., Shakhnovich E.I., Dill K.A. A Test of Lattice Protein Folding Algorithms. *Proceedings of the National Academy of Sciences*. 1995. P. 325–329.
14. Custodio F.L., Barbosa H.J., Dardenne L.E. A multiple minima genetic algorithm for protein structure prediction. *Applied Software Computing, Elsevier*. 2014. P. 88–99.
15. Gueorguiev V., Kuttel M. Implementation, Validation and Profiling of a Genetic Algorithm for Molecular Conformational Optimization. *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*. *ACM*. 2016. 16 p.
16. Mahmood A. R., Sumaiya I., Firas K., Tamjidul M.H., Abdul S. Guided macro-mutation in a graded energy based genetic algorithm for protein structure prediction. *Computational biology and chemistry*. 2016. **61**. P. 162–177.
17. Morshedian A., Razmara J., Lotfi S.. A novel approach for protein structure prediction based on estimation of distribution algorithm. *Software computing*. 2018. P. 1–12.
18. Гуляницкий Л.Ф., Рудык. В.А. Проблема предсказания структуры протеина: формализация с использованием кватернионов. *Кибернетика и системный анализ*. 2013. **49** (4). С. 130–136.

19. Nazmul R., Chetty M., Chowdhury A.R. Multimodal Memetic Framework for low-resolution protein structure prediction. *Swarm and Evolutionary Computation*. 2020. **52**. 100608.

Одержано 18.06.2020

Гуляницький Леонід Федорович,
доктор технічних наук, старший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-1379-4132

Чорножук Сергій Анатолійович,
аспірант Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
chornozhuk@gmail.com

УДК 519.8

Л.Ф. Гуляницкий, С.А. Черножук *

Генетический алгоритм с жадным стохастическим оператором скрещивания для предсказания пространственной структуры белка

Институт кибернетики имени В.М. Глушкова НАН Украины, Киев
** Переписка: chornozhuk@gmail.com*

Введение. Определение пространственной структуры белков является важной и актуальной проблемой биологии. Рассмотрев математическую модель поставленной задачи, можно сделать вывод, что она сводится к задаче комбинаторной оптимизации, а, следовательно, для нахождения решения могут быть использованы генетические и миметические алгоритмы. В статье предлагается генетический алгоритм с новым жадным стохастическим оператором скрещивания.

Цель работы. Описание генетического алгоритма с новым жадным стохастическим оператором скрещивания. В сравнении предлагаемого алгоритма с лучшими известными имплементациями генетических и миметических алгоритмов, используемых для определения пространственной структуры белка.

Результат. Работа предлагаемого алгоритма сравнивается с другими на базе 10 известных цепей длиной 48, для которых найден глобальный минимум свободной энергии, впервые предложенных в [13]. Алгоритм нашел 9 из 10 пространственных структур, на которых достигается глобальный минимум свободной энергии, а также продемонстрировал лучшее среднее значение решений, чем алгоритмы, с которыми он сравнивался.

Вывод. Экспериментально подтверждено качество работы генетического алгоритма с жадным стохастическим оператором скрещивания, потому перспективным является его дальнейшее исследования. Например, исследования подбора оптимальных параметров алгоритма, повышения быстродействия и качества найденных решений путем альтернативного кодирования.

Ключевые слова: третичная структура белка, комбинаторная оптимизация, генетические алгоритмы, оператор скрещивания, стохастичность.

UDC 519.8

Leonid Hulianytskyi, Sergii Chornozhuk *

Genetic Algorithm with New Stochastic Greedy Crossover Operator for Protein Structure Folding Problem

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
** Correspondence: chornozhuk@gmail.com*

Introduction. The spatial protein structure folding is an important and actual problem in biology. Considering the mathematical model of the task, we can conclude that it comes down to the combinatorial optimization problem. Therefore, genetic and mimetic algorithms can be used to find a solution. The article proposes a

genetic algorithm with a new greedy stochastic crossover operator, which differs from classical approaches with paying attention to qualities of possible ancestors.

The purpose of the article is to describe a genetic algorithm with a new greedy stochastic crossover operator, reveal its advantages and disadvantages, compare the proposed algorithm with the best-known implementations of genetic and memetic algorithms for the spatial protein structure prediction, and make conclusions with future steps suggestion afterward.

Result. The work of the proposed algorithm is compared with others on the basis of 10 known chains with a length of 48 first proposed in [13]. For each of the chain, a global minimum of free energy was already precalculated. The algorithm found 9 out of 10 spatial structures on which a global minimum of free energy is achieved and also demonstrated a better average value of solutions than the comparing algorithms.

Conclusion. The quality of the genetic algorithm with the greedy stochastic crossover operator has been experimentally confirmed. Consequently, its further research is promising. For example, research on the selection of optimal algorithm parameters, improving the speed and quality of solutions found through alternative coding or parallelization. Also, it is worth testing the proposed algorithm on datasets with proteins of other lengths for further checks of the algorithm's validity.

Keywords: spatial protein structure, combinatorial optimization, genetic algorithms, crossover operator, stochasticity.