УДК 004.9

ИССЛЕДОВАНИЕ МОДЕЛЬНОЙ ЗАДАЧИ СТРУКТУРНОЙ БИОИНФОРМАТИКИ

Чепурнов Андрей Владимирович[1], Ершов Николай Михайлович[2]

[1]Студент магистратуры;
МГУ им. М. В. Ломоносова,
Факультет вычислительной математики и кибернетики;
119991, Москва, ГСП-1, Ленинские горы, д. 1, стр. 52, факультет ВМК;
e-mail: andvch@ya.ru.

[2]Старший научный сотрудник;
МГУ им. М. В. Ломоносова,
Факультет вычислительной математики и кибернетики;
119991, Москва, ГСП-1, Ленинские горы, д. 1, стр. 52, факультет ВМК;
доцент;
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская область, г. Дубна, ул. Университетская, 19;
e-mail: ershovnm@gmail.com.

Работа посвящена исследованию методов решения задач структурной биоинформатики на примере решения модельной задачи укладки графов на плоскости. В работе рассматривается «энергетический» подход к решению данного типа задач, основанный на применении методов непрерывной оптимизации, целью которых является поиск конфигурации с минимумом энергии. В работе формулируется модельная задача укладки графа, формализуется структура графов, подлежащих укладке, и определяется целевая функция, моделирующая внутреннюю энергию укладки графа. Описываются нескольких популярных методов оптимизации, в том числе генетический алгоритм и алгоритм дифференциальной эволюции. Рассматривается несколько параллельных вариаций этих двух алгоритмов. Описывается реализация программной системы для автоматического тестирования заданного пользователем алгоритма при решении нескольких модельных задач фолдинга с поддержкой параллельных вычислений, веб-интерфейса и визуализации вычислений. Работа выполнена при финансовой поддержке РФФИ (грант № 20-07-01053 А).

Ключевые слова: фолдинг белка, эволюционные алгоритмы, параллельные вычисления.

Для цитирования:
Чепурнов, А. В., Ершов, Н. М. Исследование модельной задачи структурной биоинформатики // Системный анализ в науке и образовании: сетевое научное издание. – 2020. – № 3. – С. 43–52. – URL : http://sanse.ru/download/405.

STUDY OF MODEL PROBLEM OF STRUCTURAL BIOINFORMATICS

Chepurnov Andrey[1], Ershov Nikolay[2]

[1]Graduate student;
Moscow State University,
Faculty of Computational Mathematics and Cybernetics;
Russia, 119991, Moscow, GSP-1, 1-52, Leninskiye Gory;
e-mail: andvch@ya.ru.

[2]Senior research associate;
Moscow State University,
Faculty of Computational Mathematics and Cybernetics;
Russia, 119991, Moscow, GSP-1, 1-52, Leninskiye Gory;
assistant professor;
Dubna State University,
Institute of the system analysis and management;

141980, Russia, Moscow Region, Dubna, Universitetskaya str., 19;
e-mail: ershovnm@gmail.com.

The paper is devoted to the study of methods for solving problems of structural bioinformatics on the example of solving a model problem of graphs layout on a plane. The paper considers an "energy" approach to solving this type of problems, based on the use of continuous optimization methods, the purpose of which is to find a configuration with a minimum energy. The paper formulates a model problem of graph layout, describes the structure of graphs to be processed, and defines an objective function that simulates the internal energy of graph layout. Several popular optimization methods are described, including a genetic algorithm and a differential evolution algorithm. Parallel variations of these two algorithms are considered. Implementation of a software system for automatic testing of a user-defined algorithm for solving model folding problems with support for parallel computing, web interface and visualization of computations is described. The work was carried out with the financial support of the Russian Foundation for Basic Research (Grant No. 20-07-01053 A).

Keywords: protein folding, evolutionary algorithms, parallel computing.

For citation:
Chepurnov, A., Ershov, N. Study of model problem of structural bioinformatics = Исследование модельной задачи структурной биоинформатики // System Analysis in Science and Education. – № 3. – Pp. 43–52. – URL : http://sanse.ru/download/405.

Введение

Белки – это цепочки аминокислотных остатков, соединённых пептидной связью. В их строении участвуют преимущественно 20 стандартных аминокислот. Одним из этапов синтеза белка является молекулярный биологический процесс под названием трансляция, в результате которого образуется одномерная полипептидная цепь (первичная структура). Аминокислоты в ней начинают взаимодействовать друг с другом, в результате чего цепь сворачивается в уникальную объёмную структуру. Механизмы этого сворачивания (или фолдинга) плохо изучены и не могут быть качественно промоделированы. Однако информация о трехмерной структуре белка крайне важна, так как она позволяет определить функцию этого белка [1].

В результате современных широкомасштабных работ по исследованию ДНК, таких как проект «Геном человека», на сегодняшний день доступны огромные объёмы данных о первичных структурах. *GenBank* – крупнейшая база нуклеотидных и белковых последовательностей – содержит более 200 миллионов полипептидных цепей. Пространственные структуры известны всего лишь для 150 тысяч из них [2] (по состоянию на апрель 2020 года). Нахождение трехмерных структур белков экспериментальным путём трудоёмко и экономически не целесообразно.

Известно, что искомая пространственная структура имеет минимально возможную потенциальную энергию [3]. В связи с этим проблему фолдинга можно рассматривать как класс задач глобальной оптимизации. Их характерным свойством является высокая размерность и, как следствие, длительное время вычисления функции энергии. Также важно заметить, что число всевозможных конфигураций зависит от длины цепи экспоненциально. Даже для небольшого белка время работы алгоритма полного перебора превысило бы время существования вселенной. Поэтому в вычислительной биологии используются подходы, производящие поиск решений наиболее эффективно в плане количества вычислений функции энергии [4]. Существуют две стратегии: статистические методы и физические методы (методы *De novo*).

Примером статистического метода является сопоставительное моделирование (или моделирование на основании гомологии, *homology modeling*). Метод заключается в поиске среди последовательностей, для которых уже известна пространственная структура, гомологичных (или идентичных) последовательности моделируемого белка. Исходя из предположения, что белок будет иметь подобную структуру, производят построение моделей белка для гомологичных последовательностей. Только после этого вычисляют энергию, проводят оптимизацию составленных структур, выбирают наиболее удачную в качестве результата. В случае, если в базе не удалось найти гомологичные бел-

ковые последовательности, моделируемый белок «нарезают» (*threading, fold recognition*), производят процедуру для отдельных частей, а в конце оптимизируют структуру целиком.

Однако статистические методы не могут претендовать на универсальность. Поэтому для прогнозирования структуры белка предпочтительнее использовать методы *De novo*, то есть напрямую по первичной структуре, без использования каких-либо явных шаблонов. Установлено, что информации только лишь о первичной структуре должно быть достаточно для получения точного решения [3]. Крупнейший в мире проект добровольных вычислений, направленный на решение проблемы фолдинга, Rosetta@Home использует такие методы [5]. Дальнейшее развитие методов *De novo* остаётся весьма актуальным.

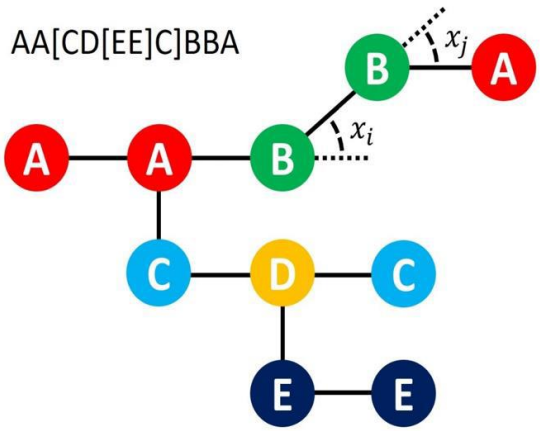
Как уже было отмечено, функция энергии белковой структуры обладает высокой вычислительной сложностью. Проводить исследования и адаптацию методов на оригинальных задачах было бы крайне неудобно из-за огромных временных затрат. В связи с этим было принято решение делать это на упрощенных модельных задачах укладки графа, сохраняющих характерные свойства исходных задач. Мы полагаем, что вводимые упрощения будут допустимы, если для последовательностей элементов аналогичных аминокислотным остаткам (имеющих аналогичные численные характеристики) решением модельной задачи будет подобная структура. Подобные идеи используются при крупнозернистом моделировании [6].

Целями настоящей работы являются:

- формализовать класс модельных задач укладки графа;
- исследовать существующие эволюционные алгоритмы и используемые в них подходы, выделить среди них пригодные для решения поставленной модельной задачи;
- реализовать выбранные алгоритмы и универсальную систему их тестирования;
- выполнить численное исследование работы рассматриваемых алгоритмов на вычислительном комплексе IBM Polus факультета ВМК МГУ.

Модельная задача

Пусть дан связный ациклический граф $G = (V, E)$ на плоскости (дерево), все вершины которого имеют одинаковый набор свойств (числовых характеристик), а все рёбра имеют одинаковую фиксированную длину. В наших исследованиях мы ограничивались двумя свойствами – условными «массой» и «зарядом» вершин, отвечающими за их взаимное притягивание и отталкивание. Каждая вершина дерева кроме корневой имеет одну и только одну родительскую, с которой соединена (входным) ребром. Таким образом, если считать фиксированными положения двух первых вершин основной цепочки, любую пространственную структуру (конфигурацию) графа можно представить $n = |E| - 2$ параметрами $x_1, x_2, \ldots, x_n$, где x_i – угол между входным ребром i-й вершины и входным ребром ее родительской вершины (рис. 1). Также пусть задана некоторая функция энергии $F(x_1, x_2, \ldots, x_n)$ заданной укладки графа, минимум которой и требуется найти.



	"масса"	"заряд"
A	1	1
B	1	-1
C	10	0
D	5	3
E	5	-3

Рис. 1. Пример укладки графа, заданного строкой AA[CD[EE]C]BBA

Граф G – это входные данные конкретной модельной задачи. Для проведения в дальнейшем вычислительных экспериментов было необходимо продумать формат представления этих данных. В качестве такого формата можно использовать специальное строковое представление. Пусть имеется алфавит символов, каждому из которому сопоставлены значения свойств вершин («масса» и «заряд»). Тогда строка из таких символов задаёт на плоскости линейный граф (цепочку) из последовательно соединённых рёбрами вершин, каждой из которых соответствует символ строки. Если после некоторого символа (кроме самого первого) участок строки заключён в квадратные скобки, будем считать, что эта подстрока задаёт подграф для вершины, соответствующий этому символу (то есть существует ребро между ней и корневой вершиной подграфа, см. пример на рис. 1). Значения свойств каждой вершины определяются соответствующим ей символом. В такой формулировке символы представляют из себя аналоги аминокислотных остатков, из которых состоят полипептидные цепи.

Функцию энергии белковой молекулы находят как сумму атомных взаимодействий в ней [5]. Аналогичным образом зададим нашу целевую функцию F:

$$F = \sum_{\substack{u,v \in V \\ u \neq v}} \left(\alpha \frac{mass(u)\, mass(v)}{r(u,v)^A} + \beta \frac{charge(u)\, charge(v)}{r(u,v)^B} \right),$$

где $r(u,v)$ – расстояние между вершинами u и v, α, β, A, B – вещественные коэффициенты. Из формулы видно, что функция F разрывна в точках, когда некоторые вершины занимают одно и то же положение в пространстве и расстояние между ними становится нулевым. В настоящих белковых структурах такие конфигурации физически невозможны. В связи с этим для вершин, расстояние между которыми меньше некоторого фиксированного числа l, вместо соответствующего слагаемого в F нужно прибавлять штрафное слагаемое (см. рис. 2).

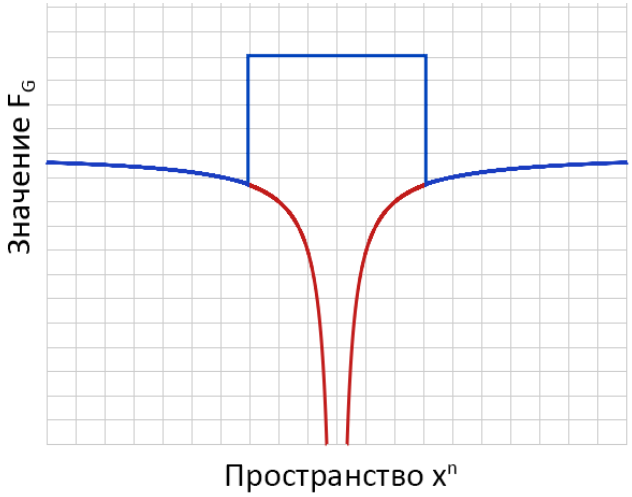


Рис. 2. Поведение функции энергии F без штрафа (красный) и со штрафом (синий)

Конкретные параметры для вычисления функции энергии подбирались уже с использованием программной реализации экспериментально. Брались графы разной длины аналогичные альфа-листам и бета-спиралям (вторичным структурам, см. рис. 3) – наиболее распространённым локальным структурам в белковых молекулах [4]. Для них вручную задавалась углы для подобных конфигураций на плоскости, которые должны быть правильными решениями модельных задач. Затем запускался простейший алгоритм, который случайным образом немного изменял эти углы. Если в этом или последующих вычислительных экспериментах обнаруживалась конфигурация с меньшей энергией, параметры функции энергии корректировались эмпирически, основываясь на особенностях этой конфигурации.

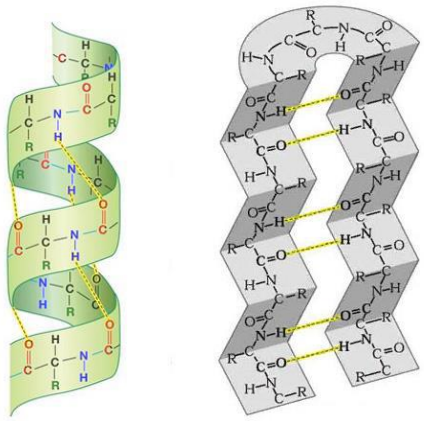


Рис. 3. Альфа-спираль (слева) и бета-лист (справа)

Алгоритмы оптимизации

Наиболее развитый класс эволюционных алгоритмов оптимизации – это генетические алгоритмы (ГА, *genetic algorithm, GA*). Кроме того, популярным эволюционным алгоритмом является алгоритм дифференциальной эволюции (ДЭ, *differential evolution, DE*). И генетические алгоритмы и алгоритм дифференциальной эволюции показывают хорошие результаты для широкого спектра задач, не требуют тщательной настройки параметров для каждого конкретного графа, а также обладают хорошим потенциалом для параллельной реализации [7]. В настоящей работе использовались классические версии указанных алгоритмов со стандартными значениями их параметров.

Очевидно, что даже простой многократный запуск эволюционного алгоритма оптимизации повышает вероятность нахождения глобального экстремума. В островной модели (*island model*) алгоритм работает параллельно на нескольких узлах вычислительной системы, а также осуществляет обмен решениями между ними. Для этого вся популяция разбивается на некоторое число субпопуляций, каждая из которых помещается на отдельный узел. Затем субпопуляции эволюционируют под действием операторов эволюционного алгоритма. Раз в несколько поколений на узлах синхронно применяется оператор миграции. Этот оператор производит обмен части субпопуляций между соседними вычислительными узлами (у которых есть прямая связь).

В клеточных эволюционных алгоритмах (*cellular evolutionary algorithm, cEA*) популяции организованы сетками, каждое решение имеет свою фиксированную клетку в них. Это позволяет накладывать ограничения на выбор пары для совместных операторов. Например, в качестве пар для отбора турнирным методом можно выбирать только решения в соседних клетках. Благодаря этому, решения, расположенные в разных частях сетки, оказываются изолированными друг от друга, что способствует сохранению разнообразия в популяции. Для клеточных алгоритмов тоже вводят оператор миграции. В этой модели решения мигрируют в соседние клетки. Также в клеточных алгоритмах возможно задавать неоднородные параметры, зависящие от положения в сетке [8]. Например, у краёв сетки параметр мутации можно установить высоким, а в центре почти нулевым. В параллельной версии клеточного алгоритма части сетки размещены на разных вычислительных узлах. Для эффективных коммуникаций соседние части сетки должны находится на узлах, между которыми есть прямая связь.

Программная реализация

Вариативность исследуемых алгоритмов предопределила необходимость разработки программного комплекса для анализа алгоритмов непрерывной оптимизации для решения задачи укладки графа на плоскости. К тестирующей системе были поставлены следующие требования:

- возможность тестирования алгоритмов, оперирующих только функцией энергии и не учитывающих фактическое представления графа;
- универсальный способ передачи параметров алгоритмов;
- автоматический сбор статистики при выполнении алгоритмов;

- визуализация получаемых результатов;
- структурированное хранение всех полученных результатов.

Реализованная система может использоваться удалённо через простой пользовательский веб-интерфейс. Также возможно взаимодействие через интерфейс командной строки. Тестируемые алгоритмы описываются на языке C++, возможно с применением технологии *Message Passing Interface* (*MPI*). На странице загрузки *upload.php* (рис. 4) пользователь вводит код алгоритма, имя теста, прописывает используемые в нём параметры и их значения, выбирает граф для тестирования и загружает эту информацию в систему.

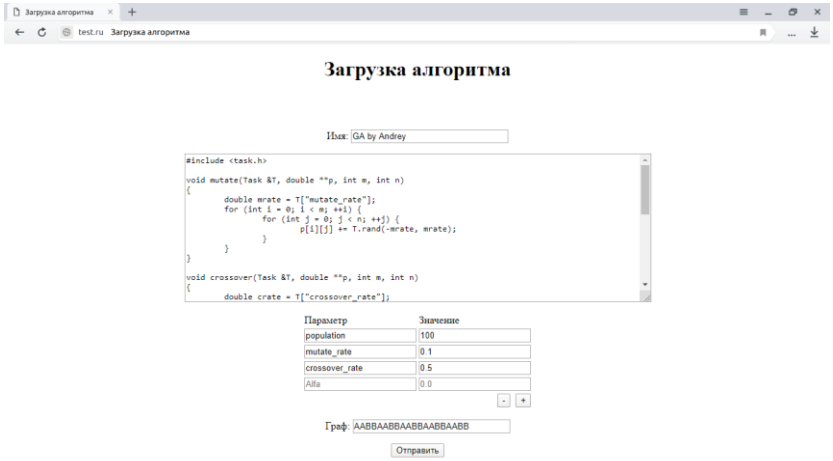


Рис. 4. Страница upload.php

Система несколько раз запускает алгоритм на всех доступных ей вычислительных узлах. Результатом работы является директория, содержащая текстовые файлы формата out<Run>_<Node>. Эти файлы содержат в каждой строке номер популяции, среднее значение функции энергии, лучшее решение и значение функции энергии для него. Затем для проведённых запусков строятся графики средних значений функции энергии по поколениям для всех узлов. Для узла, на котором было достигнуто минимальное значение функции энергии, происходит визуализация результатов.

На странице результатов *table.php* (рис. 5) можно посмотреть статус тестирования алгоритма, минимальное полученное значение функции энергии, ссылки на графики и изображения. При повторном запуске можно поменять параметры алгоритма и граф для тестирования.

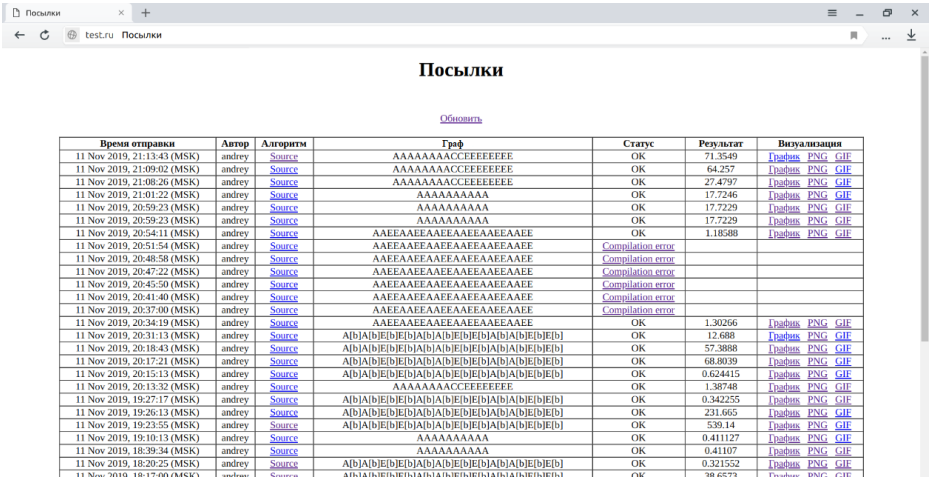


Рис. 5. Страница table.php

Принцип работы системы схематично проиллюстрирован на рис. 6. Для каждого запуска выделяется уникальный идентификатор (номер) и создаётся отдельная директория. В ней создаётся файл *alg.cpp* с кодом алгоритма, а также специальный конфигурационный файл *input* с параметрами алгоритма и информацией о самой задаче (тестовый граф, параметры функции энергии). После этого

вызывается утилита *make*, которая компилирует объектный файл и производит его линковку с объектными файлами тестирующей системы. Полученная программа запускается и получает на вход конфигурационный файл *input*.

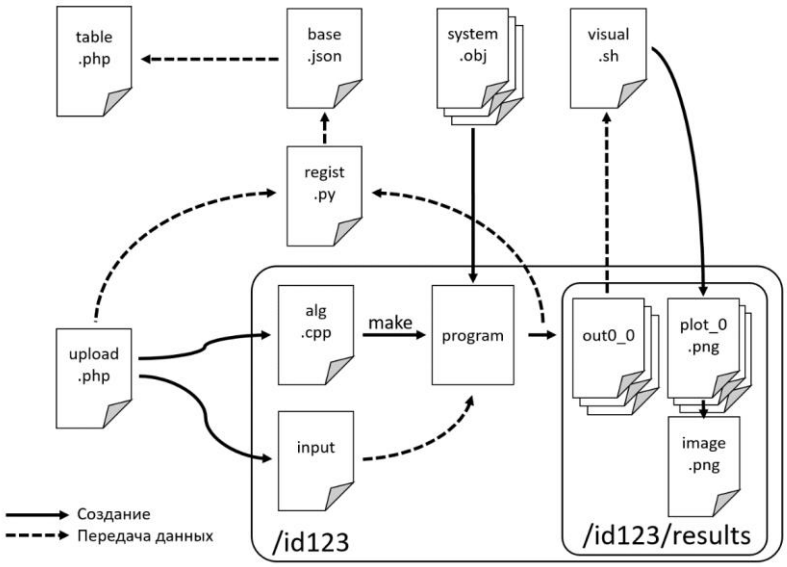


Рис. 6. Схема функционирования модуля «Добавление данных»

После успешного завершения программы выполняется сценарий, вызывающий *python*-программы для отрисовки графиков и изображений по полученным данным в директории *results*. После каждого этапа работы, а также при возникновении каких-либо ошибок, *python*-программа *regist*.py регистрирует информацию в файле *base.json*, который отображается пользователю на странице *table*.php. Помимо прочего, в *base.json* хранятся имена и время загрузки для всех тестов.

При повторном запуске компиляция уже не происходит. В новой директории создаётся символьная ссылка на программу, которая запускается с новым конфигурационным файлом *input*. Информация о повторных запусках также регистрируется в *base.json*.

Численные эксперименты

Для проведения вычислительных экспериментов тестовая система была установлена на вычислительном комплексе *IBM Polus* факультета ВМК МГУ. С целью первичной проверки работоспособности предложенных методов, алгоритмы были протестированы на полных двоичных деревьях с вершинами с одинаковым зарядом. Для них всех удалось получить структуры, в которых нет пересечений рёбер, а расстояние между близко расположенными вершинами приблизительно одинаково (см. рис. 7).

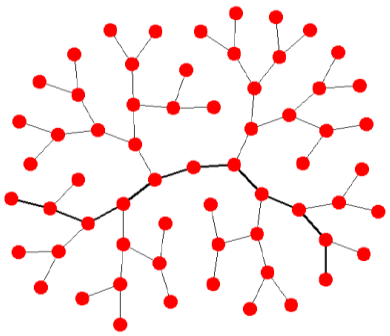


Рис. 7. Пример укладки полного двоичного дерева высоты 6

Все дальнейшие численные эксперименты проводились на графах, аналогичных альфа-спиралям, бета-листам и их комбинациям (см. рис. 8). В введённом выше строковом представлении эти графы имеют формат $A^nC^mB^n$ (при $m \ll n$), $(AABB)^n$ и $(A^nB^n)^m$ соответственно (запись X^k обозначает повторение строки X k раз), где A – лёгкие вершины с положительным зарядом, B – такие же, но с противоположным по знаку зарядом, C – тяжёлые вершины с нейтральным зарядом.

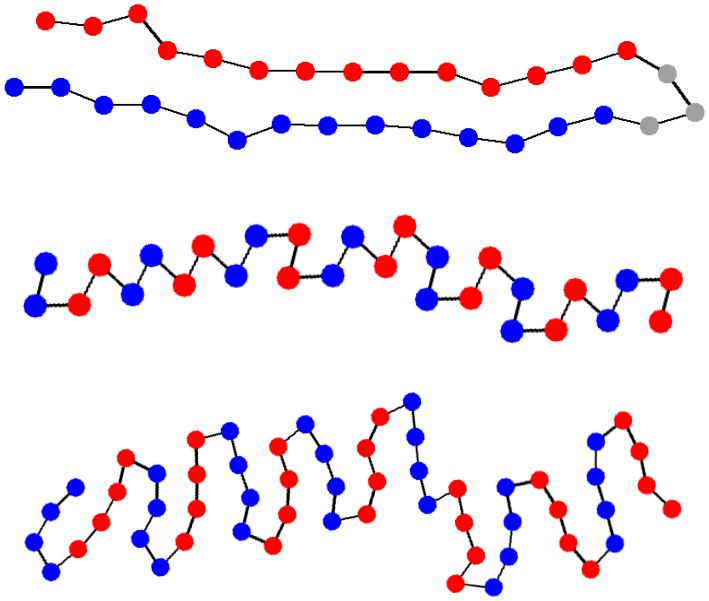


Рис. 8. Примеры укладки трех типов графов, на которых проводились численные эксперименты

Целями проведения вычислительных экспериментов была настройка параметров исследуемых алгоритмов оптимизации, а также сравнение эффективности различных вариаций генетического алгоритма и алгоритма дифференциальной эволюции. Эксперименты показали, что бета-листы – более простые структуры, чем альфа-спирали. Для небольших графов с их нахождением так или иначе справляются все вариации рассматриваемых алгоритмов. С ростом числа вершин последовательные алгоритм дифференциальной эволюции и генетический алгоритм сходятся к худшим решениям, параллельные же версии продолжают показывать неплохие результаты.

Алгоритмы дифференциальной эволюции (с островной моделью и с оператором мутации) находят альфа-спирали лучше генетических. Но даже у них в структурах часто неверно повёрнуты рёбра на локальных сегментах. Скорее всего для решения этой проблемы необходимо усложнять модель. Для длинных графов ни один алгоритм не показал удовлетворительных результатов в полной мере. Однако алгоритм дифференциальной эволюции с оператором миграции и клеточный алгоритм с неоднородным параметром мутации находят наилучшие структуры, выстраивая наиболее длинные спирали в локальных областях. Это справедливо и для более сложных структур, являющихся комбинацией альфа-спиралей и бета-листов.

Также был проведён анализ эффективности работы параллельных алгоритмов (рис. 9 и 10). В островной модели коммуникации между процессами происходят редко, поэтому её ускорение намного лучше, чем у клеточного алгоритма, в котором коммуникации происходят на каждой итерации.

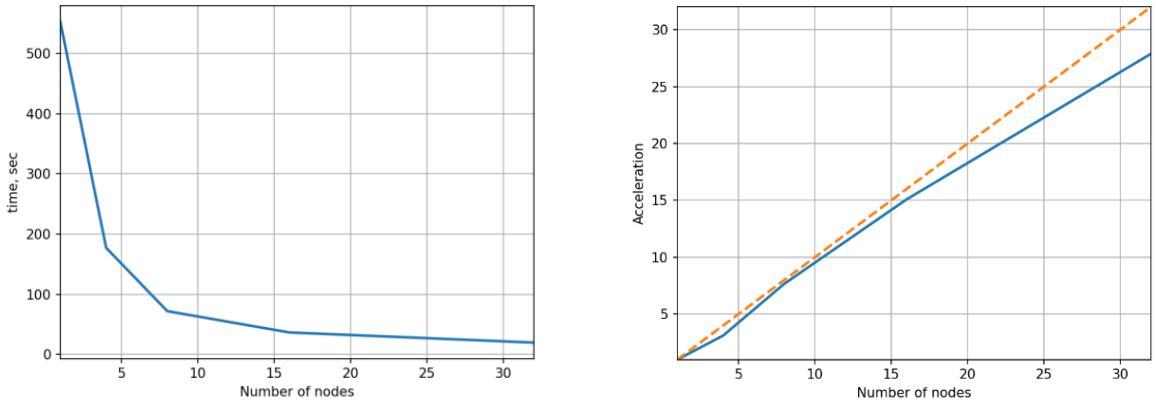


Рис. 9. Время работы и ускорение островной модели (граф из 100 вершин, размер субпопуляций – 50, коммуникации между узлами раз в 25 поколений, всего 200 000 вычислений функции энергии)

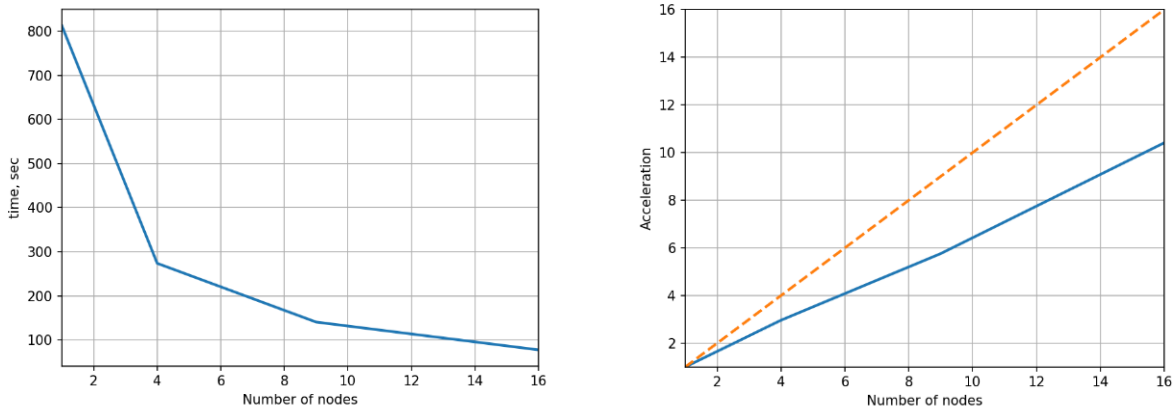


Рис. 10. Время работы и ускорение клеточного алгоритма (граф из 100 вершин, размер сетки 12×12, всего 200 000 вычислений функции энергии)

Заключение

В результате выполненной работы были получены следующие результаты:

- в результате проведённого обзора силовых алгоритмов визуализации графов и эволюционных алгоритмов, изучения теоретического обоснования и принципов их работы отобраны алгоритмы для тестирования;
- для проведения вычислительных экспериментов разработана система тестирования последовательных и параллельных алгоритмов решения задачи укладки графа на плоскости;
- в рамках этой системы выполнена реализация нескольких последовательных и параллельных эволюционных алгоритмов с использованием технологии MPI;
- анализ результатов проведенных вычислительных экспериментов показал, что наилучшую сходимость к качественному решению для наиболее широкого класса структур имеют гибрид алгоритма дифференциальной эволюции с островной моделью и клеточный алгоритм с неоднородным параметром мутации.

Список литературы

1. Whisstock, J. C., Lesk, A. M. Prediction of protein function from protein sequence and structure // Quarterly reviews of biophysics. – 2003. – №. 3. – Pp. 307–340.
2. Worldwide Protein Data Bank Deposition Statistics [HTML]. – URL : http://www.wwpdb.org/stats/deposition (дата обращения: 16.06.2020).
3. Anfinsen, C. Principles that Govern the Folding of Protein Chains // Science. – 1973. – Pp. 223–230.
4. Молекулярное моделирование: теория и практика / Х.-Д. Хельтье, В. Зиппль, Д. Роньян [и др.]. – М. : Бином. Лаборатория знаний, 2016.
5. Prediction of CASP6 structures using automated Robetta protocols / D. Chivian, D. E. Kim, L. Malmström [et al.] // Proteins: Structure, Function, and Bioinformatics. 2005. – №. S7. – Pp. 157–166.
6. Теоретические методы исследования наноструктур / О. Е. Глухова, И. В. Кириллова, И. Н. Салий [и др.] // Вестник Самарского университета. Естественнонаучная серия. – 2012. – №. 9 (100). – С. 106–117.
7. Карпенко, А. П. Современные алгоритмы поисковой оптимизации. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2014.
8. Ершов, Н. М. Неоднородные клеточные генетические алгоритмы // Компьютерные исследования и моделирование. – 2015. – №. 3. – С. 775–780.