УДК 004.89; 004.42 doi: 10.15827/0236-235X.150.337-344 2025. Т. 38. № 2. С. 337–344

Декомпозиция расчетной сетки с помощью генетического алгоритма

А.А. Рыбаков [1]✉

[1] Национальный исследовательский центр «Курчатовский институт», г. Москва, 123182, Россия

Ссылка для цитирования
Рыбаков А.А. Декомпозиция расчетной сетки с помощью генетического алгоритма // Программные продукты и системы. 2025. Т. 38. № 2. С. 337–344. doi: 10.15827/0236-235X.150.337-344
Информация о статье
Группа специальностей ВАК: 2.3.5
Поступила в редакцию: 21.01.2025 После доработки: 06.02.2025 Принята к публикации: 20.02.2025

Аннотация. Статья посвящена проблеме декомпозиции расчетных сеток для организации эффективных высокопроизводительных вычислений на многопроцессорных вычислительных системах. Проблема декомпозиции расчетной сетки рассматривается в терминах декомпозиции ее дуального графа. Для оценки качества получаемого решения используется взвешенная сумма основных показателей качества декомпозиции (отклонение размера домена от теоретического среднего значения, протяженность наибольшей границы между доменами, сумма границ между всеми парами доменов). В качестве метода декомпозиции применяется генетический алгоритм, в котором генотипом является набор опорных вершин графа для наращивания доменов. Для построения особи из генотипа используется быстрый алгоритм – грубый аналог алгоритма пузырькового роста. Эксперименты показали, что генотип небольшого размера позволяет получать достаточно качественное решение по декомпозиции расчетной сетки за приемлемое время, несмотря на грубость алгоритма генерации особи. При декомпозиции тестовой расчетной сетки величина штрафной функции лучшей особи в популяции продемонстрировала уменьшение на 50–75 % в первые 100 эпох работы алгоритма. Рассмотренный генетический алгоритм не зависит от структуры обрабатываемой расчетной сетки, прост в реализации, допускает параллельное выполнение, хорошо масштабируется на сетки больших размеров. Алгоритм применим для работы с расчетными сетками, динамически изменяющимися в процессе вычислений, так как изменение геометрии и структуры расчетной сетки слабо влияет на генотип ее декомпозиции. Он также может быть использован для декомпозиции произвольных расчетных сеток. Рассмотренный алгоритм можно расширить с помощью произвольных операций генерации декомпозиции из ее генотипа.

Ключевые слова: генетический алгоритм, расчетная сетка, декомпозиция, дуальный граф, алгоритм пузырькового роста

Благодарности. Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт» по теме FNEF-2024-0016

Введение. При выполнении вычислений на расчетных сетках с большим количеством ячеек стоит задача эффективного использования современных высокопроизводительных многопроцессорных вычислительных систем. Эффективность и масштабируемость процесса существенно зависят от качественной декомпозиции расчетной сетки. С ее помощью можно сбалансировать вычислительную нагрузку между отдельными процессорами, а также минимизировать объем обмена данными между ними [1]. Задача декомпозиции расчетной сетки сводится к задаче разбиения ее дуального графа на сбалансированные блоки с небольшим количеством ребер между ними [2]. Существует немало алгоритмов декомпозиции графов [3, 4], а также различных инструментов, включая такие широко известные продукты, как METIS/ParMETIS, Scotch/PT-Scotch, KaHyPar, Sphynx и многие другие. Большое разнообразие алгоритмов декомпозиции объясняется различными требованиями, предъявляемыми в зависимости от размера рассматриваемого графа и допустимой погрешности решения. Следует отметить, что в общем случае задача декомпозиции расчетной сетки (декомпозиции графа) является NP-полной, что практически исключает возможность нахождения ее точного решения за приемлемое время. В данной статье описаны подход и практическая реализация решения задачи декомпозиции расчетной сетки, которые опираются на автоматизированный эвристический генетический алгоритм поиска эффективного решения с использованием простого быстрого алгоритма декомпозиции и без привязки к конкретной функции оценки качества решения.

Показатели качества декомпозиции расчетной сетки

Высокопроизводительные вычисления, выполняющиеся на расчетных сетках, как правило, состоят из большого количества отдельных по времени итераций, на каждой из которых обрабатываются все ячейки сетки, а между итерациями соседние ячейки обмениваются информацией на общей границе. Сначала рассмотрим одну итерацию расчетов без учета информационных обменов. Возьмем содержащую n ячеек расчетную сетку, которую требуется декомпозировать для обработки на k-процессорном вычислительном кластере. Все ячейки расчетной сетки могут обрабатываться параллельно. Будем считать, что все ячейки являются одинаковыми с точки зрения времени их обработки. Если разбить расчетную сетку на k доменов с одинаковым количеством ячеек и обрабатывать каждый домен на отдельном процессоре, то обработка каждого домена на одной итерации будет занимать одно и то же время. Так как обработка всех доменов будет производиться параллельно, это время и станет временем, затрачиваемым на одну итерацию обработки всех ячеек сетки. Если распределение ячеек сетки по доменам неравномерное, то время выполнения одной итерации расчетов будет определяться самым крупным доменом (так как время его обработки максимально). Таким образом, в качестве показателя качества декомпозиции сетки можно принять максимальное отклонение размера домена от теоретически возможного среднего значения:

$$D = \max_{i=1,k} n_i - n/k,$$

где n_i – количество ячеек в i-м домене. В идеальном случае равномерного распределения ячеек по доменам показатель D становится равным нулю.

После завершения итерации расчетов требуется произвести информационные обмены данными между парами соседних ячеек. Если ячейки находятся в одном домене, то такой обмен не представляет проблемы. Если ячейки относятся к разным доменам, то информационный обмен осуществляется с применением механизмов межпроцессного взаимодействия (например, с использованием MPI [5]). Таким образом, для каждой пары доменов, имеющих общую границу, необходимо организовывать межпроцессный обмен. Все такие обмены могут происходить одновременно, и общее затрачиваемое на них время определяется длиной максимальной общей границы между доменами. Для учета информационных обменов рассмотрим дуальный граф расчетной сетки, вершинами которого являются ячейки сетки. Две вершины в дуальном графе соединены ребром, если две соответствующие ячейки соседние (имеют общую грань). Множество ребер дуального графа обозначим через E, а для каждого отдельного ребра e под e_a и e_b будем понимать инцидентные ему вершины. Тогда в качестве второй характеристики качества декомпозиции используем величину наиболее протяженной границы между парой доменов, или

$$L_{ij} = \left|\left\{e \in E : \left\{d(e_a), d(e_b)\right\} = \left\{i, j\right\}\right\}\right|,$$

$$L = \max_{1 \le i < j \le k} L_{ij},$$

где $d(v)$ – домен, к которому относится вершина v. В идеальном случае значение характеристики L может быть сколь угодно малым, даже нулевым (в случае, если сетка представляет собой k одинаковых по количеству ячеек несвязных областей).

В роли дополнительной характеристики качества декомпозиции можно использовать суммарную длину границ между доменами:

$$I = \sum_{1 \le i < j \le j} L_{ij},$$

что соответствует общему количеству пересылаемых данных в рамках межпроцессного обмена.

Для каждой задачи перечисленные характеристики качества декомпозиции могут иметь разную важность. Как правило, обработка ячеек занимает основное время высокопроизводительных расчетов и параметр D более критичен. Однако в некоторых случаях вычислительная процедура настолько легковесная, что основные расходы приходятся на межпроцессные обмены. Например, в работе [6] описан алгоритм сглаживания поверхностной расчетной сетки, для которого при распараллеливании по MPI расходы на информационные обмены значительно превышают расходы на вычисления внутри ячеек. В дальнейшем характеристика I не рассматривается, а вместо первых двух характеристик используется единый взвешенный показатель качества декомпозиции $Q = \delta D + \lambda L$ с произвольными значениями весов δ и λ.

Использование генетических алгоритмов для решения задач комбинаторной оптимизации

Генетические алгоритмы представляют собой природоподобные эвристические алгоритмы, с помощью которых можно искать решения оптимизационных задач с большим количеством параметров [7, 8]. Во время их применения моделируются процессы создания, выживания и размножения популяции потенциально возможных решений поставленной проблемы в условиях сложно устроенной окружающей среды.

Применимость генетических алгоритмов базируется на принципе выживания сильнейших особей в популяции [9]. Работа алгоритма начинается с создания стартовой популяции потенциальных решений проблемы, каждое из которых представлено отдельной особью, а каждая особь определяется набором характеризующих ее генов (генотипом). Для особи должна быть определена функция пригодности, являющаяся индикатором ее качества для решения поставленной задачи (вместо функции пригодности можно использовать штрафную функцию, наоборот, являющуюся показателем непригодности). Генетический алгоритм итерационный. На каждой итерации по определенным законам на основе текущей популяции создается следующая популяция, которая, по ожиданиям, в среднем должна быть более приспособлена к условиям окружающей среды (то есть лучше удовлетворять решению поставленной задачи). Процесс получения новых популяций продолжается до тех пор, пока не будет достигнуто некоторое условие (либо получено решение требуемого качества, либо дальнейшее улучшение не наблюдается) [10].

Генетические алгоритмы широко применяются во многих областях для решения задач с большим количеством параметров и ограничений. Конечно, они не гарантируют отыскание оптимального варианта, однако с их использованием можно быстро найти достаточно качественное в текущих условиях решение. Генетические алгоритмы применимы для задач планирования [11] и эффективного использования ресурсов [12, 13]. Они применяются для анализа данных, поиска тенденций и предсказаний, где не могут быть получены точные решения [14, 15]. Одна из областей активного применения генетических алгоритмов – задачи комбинаторной оптимизации [16]. Точное решение таких задач зачастую не может быть получено из-за большого количества параметров и отсутствия полиномиального алгоритма для поиска ответа. Генетические алгоритмы позволяют находить приемлемые решения для таких классических задач, как задача коммивояжера [17], задача нахождения оптимального пути [18], раскраска графа [19], декомпозиция графа [20].

При использовании генетических алгоритмов важно различать понятия генотипа и особи. Генотип – это набор генов, некий код механизма создания особи. Особь – сформированный на основе генотипа индивид, обладающий определенными свойствами и характеристиками, для которого может быть вычислена функция пригодности или штрафная функция. Зачастую при использовании генетических алгоритмов различие между генотипом и особью стирается и вместо генотипа используется просто явное представление особи. Так, в работе [20] генотип представляет собой точное соотнесение каждой вершины графа и результирующего домена. Использование генотипа в роли особи в генетических алгоритмах приводит к существенному замедлению сходимости, что снижает ценность этих алгоритмов. К тому же применение механизмов скрещивания и мутаций к особям вместо генотипов вызывает сомнение с точки зрения корректности таких операций. В данной работе в качестве генотипа используется максимально короткий код, по которому может быть быстро построена особь.

Генетический алгоритм декомпозиции расчетной сетки

Рассмотрим алгоритм декомпозиции дуального графа произвольной расчетной сетки на k доменов. Количество вершин в графе равно n. При этом рассматриваемый дуальный граф задан структурами inc и es. В структуре inc хранятся соседи каждой вершины графа: inc[i] – вектор номеров всех вершин, смежных i-й вершине. Структура es – множество ребер графа.

В качестве особи рассмотрим вектор colors длины n, где colors[i] – номер домена, к которому отнесена i-я вершина графа. В качестве функции качества особи используем функцию $Q = \delta D + \lambda L$ (чем ниже значение данной функции, тем выше качество особи) с весами $\delta = 10$ и $\lambda = 1$. Исходя из этого, функцию Q можно называть штрафной функцией особи. Под популяцией будем понимать набор отдельных особей.

Ключевыми элементами генетического алгоритма являются понятие генотипа и процесс формирования особи на его основе. В качестве генотипа используем вектор genotype длины k, задающий k номеров вершин дуального графа, которые будем называть опорными вершинами. При инициализации каждую опорную вершину genotype[i] будем принудительно относить к i-му домену. При построении особи на основе генотипа распределяем вершины между доменами с помощью простого алгоритма, схожего с алгоритмом пузырькового роста, начиная с опорных вершин и обходя граф в ширину. Представим программный код процедуры распределения вершин между доменами:

```
deque<size_t> q;

for (auto i{ 0 }; i < k; ++i)
{
    colors[genotype[i]] = i;
    q.push_back(genotype[i]);
}

while (!q.empty())
{
    auto node{ q.front() };
    auto color{ colors[node] };
    q.pop_front();

    for (auto ngh : inc[node])
    {
        if (colors[ngh] == -1)
        {
            colors[ngh] = color;
            q.push_back(ngh);
        }
    }
}
```

Приведенный алгоритм декомпозиции по опорным вершинам не является сколь угодно оптимальным (и даже намеренно наиболее простой) и создает несбалансированные разбиения, однако он быстрый, что позволяет создавать большое количество особей за короткое время.

На основе выбранного простого способа построения особи моделируем процесс эволюции следующим образом. Пусть имеется текущая популяция, в которой для каждой особи известна характеристика ее качества (она вычисляется при построении особи и не меняется на протяжении всей ее жизни). На первой фазе удаляем из популяции некоторое количество наихудших особей. Из оставшейся популяции случайно сформированные пары особей на основе операции скрещивания (кроссовера) будут производить и добавлять потомство в популяцию для восстановления ее исходного размера. Операция скрещивания устроена предельно просто: для каждого номера i от 1 до k в качестве i-го гена случайным образом выбирается i-й ген одного из родителей. После выполнения скрещивания в новой особи с заданной вероятностью применяется мутация (рис. 1).

Так как ген представляет собой номер опорной вершины дуального графа, в качестве мутации выбрана операция замены случайной опорной вершины на выбранную случайным образом смежную с ней вершину (рис. 2).

Описанный генетический алгоритм поиска эффективной декомпозиции характеризуется следующими глобальными параметрами: количество эпох– количество применяемых глобальных операций по вымиранию части популяции и ее восстановлению; размер популяции (population_size) – количество особей в популяции; доля вымирания (extinction_ratio) – доля наименее эффективных особей, вымирающих на каждой эпохе; вероятность мутации – возможность мутации в новой особи.

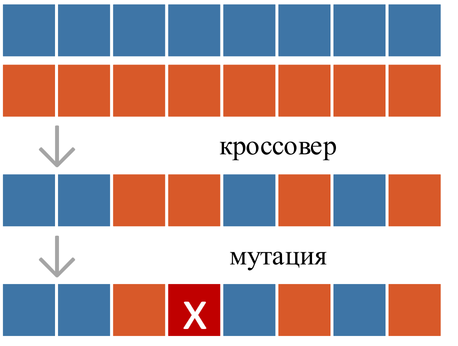

Рис. 1. Схема кроссовера и мутации новой особи

Fig. 1. Crossover and mutation scheme of a new individual

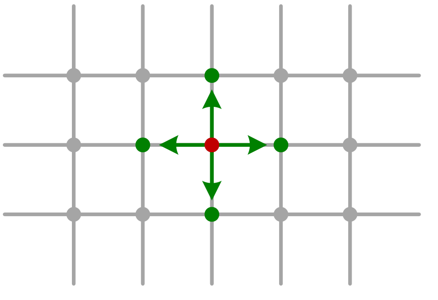

Рис. 2. Мутация особи путем движения опорной вершины к одной из ее смежных вершин

Fig. 2. Individual mutation by movement of a reference vertex to one of its adjacent vertices

Эксперимент по применению генетического алгоритма

Применим предложенный генетический алгоритм для решения задачи декомпозиции двумерной прямоугольной структурированной расчетной сетки размером 100×100 ячеек. Такой вид сетки выбран для удобства визуализации результатов, сам алгоритм применим для расчетных сеток произвольного вида, поскольку на вход принимает только информацию о структуре дуального графа.

Количество эпох выбрано равным 500. Алгоритм прекращает работу либо после обработки последней эпохи, либо после того, как все особи в популяции сравняются по характеристике эффективности.

Размер популяции выбран равным 50. При его выборе необходимо придерживаться баланса, так как при слишком большом значении этого параметра время работы алгоритма существенно возрастает, а при слишком маленьком ограничивает разнообразие особей в популяции, что приводит к ранней остановке алгоритма в локальном минимуме, где все особи обладают одинаковым, достаточно плохим показателем эффективности.

Доля вымирания принята равной 0,2. При выборе доли вымирания не стоит использовать слишком высокие значения, поскольку это приводит к излишне интенсивному отсеву особей, что может вызвать преждевременную отбраковку средних по эффективности особей, способных дать более эффективное потомство.

Вероятность возникновения мутации принята равной 0,2. Поскольку мутация представляет собой смещение опорной вершины по ребру к одному из своих соседей (что является достаточно незначительным изменением), такое частое возникновение мутаций оправданно.

Типичные графики значения штрафной функции лучшей особи в популяции в зависимости от номера эпохи приведены на рисунке 3. Эксперименты показали, что при выбранном наборе макропараметров алгоритма примерно в первые 100 эпох наблюдается существенное улучшение лучшей особи в популяции, затем медленное улучшение наблюдается еще около 150 эпох, после чего работа алгоритма останавливается в локальном минимуме при заполнении всей популяции копиями лучшей особи. При этом значение штрафной функции лучшей особи за время работы алгоритма падает примерно на 50–75 %, если считать от значения лучшей особи стартовой популяции. Для повышения качества работы алгоритма можно включить механизм предотвращения его ранней остановки. Для этого при возникновении копий лучшей особи можно применять к ней так называемые макромутации, что соответствует принудительному выталкиванию особи из локального минимума штрафной функции [21]. В данной работе этот механизм не применялся.

На рисунке 4 приведены характерные варианты изменения лучшей особи при переходе от текущей популяции к популяции следующей эпохи. В верхней части рисунка проиллюстрирована ситуация, когда одна часть опорных вершин лучшей особи сохранилась, а другая была заменена на опорные вершины второго родителя, что в целом привело к снижению значения штрафной функции. В нижней части показано влияние последовательности мутаций на лучшую особь. Так, даже незаметное на рисунке смещение опорных вершин привело к визуально различимому изменению геометрии образованных этими опорными вершинами доменов.

Следует отметить, что, хотя реализация генетического алгоритма и подразумевает посто-

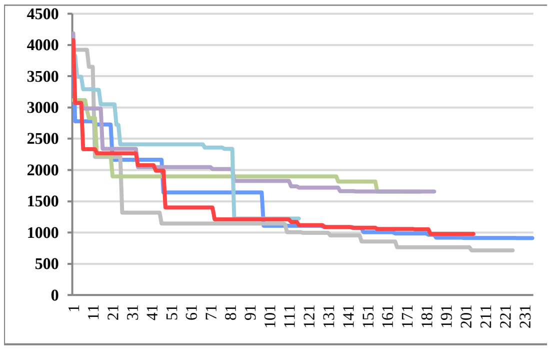

Рис. 3. Характерные графики штрафной функции лучшей особи в популяции в зависимости от номера эпохи

Fig. 3. Typical graphs of the penalty function of the best individual in the population depending on the epoch number

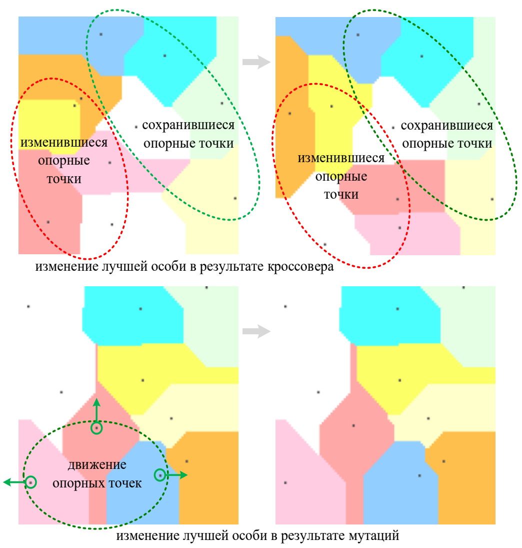

Рис. 4. Замещение части генотипа в процессе скрещивания (сверху) и мутация опорных вершин (снизу)

Fig. 4. Partial genotype replacement during crossover (top) and base vertex mutation (bottom)

янное улучшение эволюционирующей популяции, это никак не гарантируется. К тому же при скрещивании двух особей нет гарантии, что их потомство будет более приспособленным к поставленной задаче и что применяемые мутации всегда приведут к более качественному решению. Поэтому возникает вопрос, не является ли сам факт появления удачных особей случайным и не было бы более эффективным просто сгенерировать большое количество случайных особей, после чего выбрать из них лучшего представителя. Для проверки этого предположения были проведены следующие эксперименты.

Пусть имеется одиночный запуск генетического алгоритма с макропараметрами population_size и extinction_ratio. Если в процессе работы алгоритма до его завершения прошло ecount эпох (не считая стартовой), то всего было создано icount = population_size × (1 + ecount × extinction_ratio) особей. Поэтому после завершения каждого запуска генетического алгоритма его результат сравнивался со значением лучшей особи из случайной популяции размера icount.

По результатам такого сравнения построены графики (http://www.swsys.ru/uploaded/image/2025-2/13.jpg). Красным цветом отмечен график качества лучшей особи в начале работы алгоритма, зеленым – в конце. Черным цветом отмечено значение лучшей особи случайной популяции размера icount. Также на графиках отмечены линии средних значений для серии запусков генетического алгоритма. Расчеты показали, что лучшая особь популяции генетического алгоритма опережает лучшую особь случайной популяции примерно в 1,5 раза.

Реализация описанного в статье генетического алгоритма декомпозиции расчетной сетки доступна в открытом репозитории https://github.com/r-aax/mendel.

Заключение

В статье рассмотрена задача декомпозиции расчетной сетки с произвольной функцией качества декомпозиции. Она эквивалентна задаче декомпозиции дуального графа. Для поставленной задачи описан автоматизированный эвристический генетический алгоритм. В качестве генотипа в нем использован набор опорных вершин, на основе которых выполнялась декомпозиция с помощью грубого аналога алгоритма пузырькового роста. Путем экспериментов для генетического алгоритма выбраны такие макропараметры, как ограничение на количество эпох, размер популяции, доля вымирания популяции, вероятность возникновения мутации. Эксперименты продемонстрировали применимость генетического алгоритма к задаче декомпозиции расчетной сетки. Замечено, что лучшая особь популяции за время работы алгоритма демонстрирует улучшение на 50–75 %.

По результатам экспериментов можно сделать следующие выводы относительно дальнейшего развития алгоритма. Добавление других алгоритмов построения декомпозиции по набору опорных вершин (генотипу) способно повысить качество результата генетического алгоритма. Предотвратить остановку в локальном минимуме можно с помощью введения в него макромутаций. В рассмотренном алгоритме есть неиспользованные возможности при реализации процедуры скрещивания особей: в текущей реализации скрещивания происходит случайное перемешивание опорных вершин, что с большой вероятностью не приведет к улучшению решения.

Разработанный генетический алгоритм применим к расчетным сеткам произвольного вида, с его помощью можно быстро получать качественное решение декомпозиции, произвольным образом задавая требуемую штрафную функцию решения. Алгоритм допускает параллельное исполнение, хорошо масштабируется для работы с расчетными сетками большего размера и может быть расширен с помощью других процедур генерации особи из генотипа.

Список литературы

1. Ushijima-Mwesigwa H., Hyman J., Hagberg A. et al. Multilevel graph partitioning for three-dimensional discrete fracture network flow simulations. Mathematical Geosciences, 2021, vol. 53, pp. 1699–1724. doi: 10.1007/s11004-021-09944-y.

2. Eyubov K., Faraj M., Schulz C. FREIGHT: Fast streaming hypergraph partitioning. Algorithmica, 2025, vol. 87, pp. 405–428. doi: 10.1007/s00453-024-01291-8.

3. Ayall T.A., Liu H., Zhou C., Seid A.M., Gereme F.B., Abishu H.N., Yacob Y.H. Graph computing systems and partitioning techniques: A survey. IEEE Access, 2022, vol. 10, pp. 118523–118550. doi: 10.1109/ACCESS.2022.3219422.

4. Головченко Е.Н. Обзор алгоритмов декомпозиции графов // Препринты ИПМ им. М.В. Келдыша. 2020. № 2. 38 с. doi: 10.20948/prepr-2020-2.

5. Al-Johany N.A., Sharaf S.A., Eassa F.E., Alnanih R.A. Static analysis techniques for fixing software defects in MPI-based parallel programs. CMC, 2024, vol. 79, no. 2, pp. 3139–3173. doi: 10.32604/cmc.2024.047392.

6. Tong X., Thompson D., Arnoldus Q., Collins E., Luke E. Three-dimensional surface evolution and mesh deformation for aircraft icing applications. J. of Aircraft, 2016, vol. 54, pp. 1–17. doi: 10.2514/1.C033949.

7. Katoch S., Chauhan S.S., Kumar V. A review on genetic algorithm: Past, present, and future. Multimedia Tools and Applications, 2021, vol. 80, pp. 8091–8126. doi: 10.1007/s11042-020-10139-6.

8. Wirayanti N., Sriwindono H. Implementation of hybrid genetic algorithm for solving the teacher placement problem. SSHJ, 2025, vol. 9, no. 01, pp. 6341–6347. doi: 10.18535/sshj.v9i01.1460.

9. Waysi D., Ahmed B.T., Ibrahim I.M. Optimization by nature: A review of genetic algorithm techniques. IJCS, 2025, vol. 14, no. 1, pp. 268–284. doi: 10.33022/ijcs.v14i1.4596.

10. Charilogis V., Tsoulos I.G. Introducing a parallel genetic algorithm for global optimization problems. AppliedMath, 2024, vol. 4, no. 2, pp. 709–730. doi: 10.3390/appliedmath4020038.

11. Dawei Q. Using genetic algorithm to optimize the training plan and game strategy of basketball players. Scalable Computing: Practice and Experience, 2025, vol. 26, no. 1, pp. 441–449. doi: 10.12694/scpe.v26i1.3845.

12. Fang Q. Optimization of multi-objective crop planning strategies based on genetic algorithms. Highlights in Science, Engineering and Technology, 2025, vol. 133, pp. 68–75. doi: 10.54097/7qv57m30.

13. Mahmood H.A., Al-Fatlawi O.F., Al-Janabi M.A., Sadeq D.J., Al-Jumaah Y.M., Khan M. Optimizing well placement with genetic algorithms: A case study. J. of Petroleum Research and Studies, 2024, vol. 14, no. 4, pp. 37–51. doi: 10.52716/jprs.v14i4.895.

14. Kangra K., Singh J. A genetic algorithm-based feature selection approach for diabetes prediction. IJ-AI, 2024, vol. 13, no. 2, pp. 1489–1498. doi: 10.11591/ijai.v13.i2.pp1489-1498.

15. Sangeetha V., Vaneeta M., Mamatha A., Shoba M., Deepa S.R., Sujatha V., Sujatha S. Breast cancer prediction using genetic algorithm and sand cat swarm optimization algorithm. IJEECS, 2025, vol. 37, no. 2, pp. 849–858. doi: 10.11591/ijeecs.v37.i2.pp849-858.

16. Hamdan A., Nah S.S., Leng G.S., Leng C.K., King T.W. Recent evolutionary algorithm variants for combinatorial optimization problem. Applications of Modelling and Simulations, 2023, vol. 7, pp. 214–238.

17. Kralev V., Kraleva R. Combining genetic algorithm with local search method in solving optimization problems. Electronics, 2024, vol. 13, no. 20, art. 4126. doi: 10.3390/electronics13204126.

18. Богданов М.Д., Рудинский И.Д. Использование генетического алгоритма для решения задачи поиска оптимального пути // Вестн. науки и образования Северо-Запада России. 2023. Т. 9. № 2. С. 76–88.

19. Malhotra K., Vasa K., Chaudhary N., Vishnoi A., Sapra V. A solution to graph coloring problem using genetic algorithm. EAI Endorsed Transactions on Scalable Information Systems, 2024, vol. 11, no. 6, pp. 1–8. doi: 10.4108/eetsis.5437.

20. Li M., Cui H., Zhou C., Xu S. GAP: Genetic algorithm based large-scale graph partition in heterogeneous cluster. IEEE Access, 2020, vol. 8, pp. 144197–144204. doi: 10.1109/ACCESS.2020.3014351.

21. Baranov A., Aladyshev O., Bragin K. Job mapping cyclic composite algorithm for supercomputer resource manager. In: LNCS. Proc. RuSCDays, 2025, vol. 15406, pp. 377–389. doi: 10.1007/978-3-031-78459-0_27.

Software & Systems doi: 10.15827/0236-235X.150.337-344 2025, 38(2), pp. 337–344

Computational mesh decomposition using a genetic algorithm

Aleksey A. Rybakov [1]✉

[1] National Research Centre “Kurchatov Institute”,
Moscow, 123182, Russian Federation

For citation

Rybakov, A.A. (2025) ‘Computational mesh decomposition using a genetic algorithm’, *Software & Systems*, 38(2), pp. 337–344 (in Russ.). doi: 10.15827/0236-235X.150.337-344

Article info

Received: 21.01.2025 After revision: 06.02.2025 Accepted: 20.02.2025

Abstract. The paper focuses on the problem of computational mesh decomposition for organizing efficient high-performance computations on multiprocessor computing systems. The author considers the problem of computational mesh decomposition in terms of its dual graph decomposition. To assess the quality of the obtained solution, the author uses the weighted sum of the main decomposition quality indicators (deviation of the domain size from a theoretical average value, the length of the largest boundary between domains, the sum of boundaries between all domain pairs). A decomposition method is a genetic algorithm with the genotype consisting of graph base vertices to build up domains. The author uses a fast algorithm, which is a rough analog of the bubble growth algorithm, to build an individual from a genotype. Experiments show that a small size genotype allows obtaining a sufficiently high-quality solution for decomposing the computational mesh in an acceptable time, despite the coarseness of the algorithm for generating the individual. When decomposing the test computational mesh, the value of the penalty function of the population best individual showed a 50–75 % decrease in the first 100 algorithm epochs. The considered genetic algorithm does not depend on the computational mesh structure, is simple to implement, allows parallel execution, and scales well to large meshes. The algorithm is applicable for working with computational meshes that dynamically change during computations, since changing the computational mesh geometry and structure weakly affects the genotype of its decomposition. It can also be used to decompose arbitrary computational meshes. The considered algorithm can be extended with arbitrary operations of generating a decomposition from its genotype.

Keywords: genetic algorithm, computational mesh, decomposition, dual graph, bubble growth algorithm

Acknowledgements. The work was performed within the framework of the state assignment of Research Center "Kurchatov Institute", topic FNEF-2024-0016

References

1. Ushijima-Mwesigwa, H., Hyman, J., Hagberg, A. et al. (2021) 'Multilevel graph partitioning for three-dimensional discrete fracture network flow simulations', *Mathematical Geosciences*, 53, pp. 1699–1724. doi: 10.1007/s11004-021-09944-y.
2. Eyubov, K., Faraj, M., Schulz, C. (2025) 'FREIGHT: Fast streaming hypergraph partitioning', *Algorithmica*, 87, pp. 405–428. doi: 10.1007/s00453-024-01291-8.
3. Ayall, T.A., Liu, H., Zhou, C., Seid, A.M., Gereme, F.B., Abishu, H.N., Yacob, Y.H. (2022) 'Graph computing systems and partitioning techniques: A survey', *IEEE Access*, 10, pp. 118523–118550. doi: 10.1109/ACCESS.2022.3219422.
4. Golovchenko, E.N. (2020) 'Survey of graph partitioning algorithms', *Keldysh Institute Preprints*, (2), 38 p. (in Russ.). 10.20948/prepr-2020-2.
5. Al-Johany, N.A., Sharaf, S.A., Eassa, F.E., Alnanih, R.A. (2024) 'Static analysis techniques for fixing software defects in MPI-based parallel programs', *CMC*, 79(2), pp. 3139–3173. doi: 10.32604/cmc.2024.047392.
6. Tong, X., Thompson, D., Arnoldus, Q., Collins, E., Luke, E. (2016) 'Three-dimensional surface evolution and mesh deformation for aircraft icing applications', *J. of Aircraft*, 54, pp. 1–17. doi: 10.2514/1.C033949.
7. Katoch, S., Chauhan, S.S., Kumar, V. (2021) 'A review on genetic algorithm: Past, present, and future', *Multimedia Tools and Applications*, 80, pp. 8091–8126. doi: 10.1007/s11042-020-10139-6.
8. Wirayanti, N., Sriwindono, H. (2025) 'Implementation of hybrid genetic algorithm for solving the teacher placement problem', *SSHJ*, 9(01), pp. 6341–6347. doi: 10.18535/sshj.v9i01.1460.
9. Waysi, D., Ahmed, B.T., Ibrahim, I.M. (2025) 'Optimization by nature: A review of genetic algorithm techniques', *IJCS*, 14(1), pp. 268–284. doi: 10.33022/ijcs.v14i1.4596.
10. Charilogis, V., Tsoulos, I.G. (2024) 'Introducing a parallel genetic algorithm for global optimization problems', *AppliedMath*, 4(2), pp. 709–730. doi: 10.3390/appliedmath4020038.
11. Dawei, Q. (2025) 'Using genetic algorithm to optimize the training plan and game strategy of basketball players', *Scalable Computing: Practice and Experience*, 26(1), pp. 441–449. doi: 10.12694/scpe.v26i1.3845.
12. Fang, Q. (2025) 'Optimization of multi-objective crop planning strategies based on genetic algorithms', *Highlights in Science, Engineering and Technology*, 133, pp. 68–75. doi: 10.54097/7qv57m30.
13. Mahmood, H.A., Al-Fatlawi, O.F., Al-Janabi, M.A., Sadeq, D.J., Al-Jumaah, Y.M., Khan, M. (2024) 'Optimizing well placement with genetic algorithms: A case study', *J. of Petroleum Research and Studies*, 14(4), pp. 37–51. doi: 10.52716/jprs.v14i4.895.
14. Kangra, K., Singh, J. (2024) 'A genetic algorithm-based feature selection approach for diabetes prediction', *IJ-AI*, 13(2), pp. 1489–1498. doi: 10.11591/ijai.v13.i2.pp1489-1498.
15. Sangeetha, V., Vaneeta, M., Mamatha, A., Shoba, M., Deepa, S.R., Sujatha, V., Sujatha, S. (2025) 'Breast cancer prediction using genetic algorithm and sand cat swarm optimization algorithm', *IJEECS*, 37(2), pp. 849–858. doi: 10.11591/ijeecs.v37.i2.pp849-858.
16. Hamdan, A., Nah, S.S., Leng, G.S., Leng, C.K., King, T.W. (2023) 'Recent evolutionary algorithm variants for combinatorial optimization problem', *Applications of Modelling and Simulations*, 7, pp. 214–238.
17. Kralev, V., Kraleva, R. (2024) 'Combining genetic algorithm with local search method in solving optimization problems', *Electronics*, 13(20), art. 4126. doi: 10.3390/electronics13204126.
18. Bogdanov, M.D., Rudinskiy, I.D. (2023) 'Using a genetic algorithm for solving the problem of searching the optimal path', *J. of Science and Education of North-West Russia*, 9(2), pp. 76–88 (in Russ.).
19. Malhotra, K., Vasa, K., Chaudhary, N., Vishnoi, A., Sapra, V. (2024) 'A solution to graph coloring problem using genetic algorithm', *EAI Endorsed Transactions on Scalable Information Systems*, 11(6), pp. 1–8. doi: 10.4108/eetsis.5437.
20. Li, M., Cui, H., Zhou, C., Xu, S. (2020) 'GAP: Genetic algorithm based large-scale graph partition in heterogeneous cluster', *IEEE Access*, 8, pp. 144197–144204. doi: 10.1109/ACCESS.2020.3014351.
21. Baranov, A., Aladyshev, O., Bragin, K. (2025) 'Job mapping cyclic composite algorithm for supercomputer resource manager', in *LNCS. Proc. RuSCDays*, 15406, pp. 377–389. doi: 10.1007/978-3-031-78459-0_27.

Авторы
Рыбаков Алексей Анатольевич [1],
к.ф.-м.н., начальник отдела,
rybakov@jscc.ru

[1] Национальный исследовательский центр
«Курчатовский институт»,
г. Москва, 123182, Россия

Authors
Aleksey A. Rybakov [1],
Cand. of Sci. (Physics and Mathematics),
Head of Department, rybakov@jscc.ru

[1] National Research Centre
"Kurchatov Institute",
Moscow, 123182, Russian Federation