ТЕХНИЧЕСКИЕ НАУКИ

УДК 681.5

Гибридный подход к решению NP-сложных задач настройки производства: объединение метода ветвей и границ с генетическим алгоритмом

И.И. Горовых, С.Н. Горовых
Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация

Аннотация
Оптимизация планирования задач настроек в производственном канале является неотъемлемым элементом гибкости производства и ключевым фактором повышения производительности. В данной статье рассматривается задача оптимизации производства, основанная на промышленном примере. Предлагается гибридный подход, объединяющий метод ветвей и границ с генетическим алгоритмом для решения NP-сложных задач настройки производства. Основная идея состоит в использовании генетического алгоритма для улучшения верхней границы, что позволяет ускорить процесс ветвления. Параллельно с этим генетический алгоритм использует информацию о состоянии стека ветвей и границ для сужения пространства поиска. Оба метода работают взаимодополняющим образом. Предложенный гибридный подход позволяет достичь более эффективного и точного решения NP-сложных задач оптимизации производства. Результаты исследований могут быть применены в различных индустриальных секторах, где важна эффективность и гибкость производства.

Ключевые слова: метод ветвей и границ, NP-сложные задачи, генетический алгоритм, оптимизация генетического алгоритма, теория расписания, гибридизация генетического алгоритма

Для цитирования. Горовых И.И., Горовых С.Н. Гибридный подход к решению NP-сложных задач настройки производства: объединение метода ветвей и границ с генетическим алгоритмом. *Молодой исследователь Дона.* 2024;9(1):10–19.

A Hybrid Approach to Solving NP-hard Production Problems: Combination of the Branch and Boundary Method with a Genetic Algorithm

Ivan I. Gorovykh, Svetlana N. Gorovykh
Don State Technical University, Rostov-on-Don, Russian Federation

Abstract
Optimizing the scheduling of settings tasks in the production channel is an integral element of production flexibility and is a key factor for increasing productivity. This article discusses a production optimization problem based on an industrial example. The authors propose a hybrid approach combining branch and bound method with a genetic algorithm to solve NP-hard production problems. The basic idea is to use a genetic algorithm to improve the upper bound, thereby speeding up the branching process. In parallel, the genetic algorithm uses information about the state of the branch and bound stack to narrow the search space. Both methods work in a complementary manner. The proposed hybrid approach allows us to achieve a more efficient and accurate solution to NP-hard production optimization problems. The research results can be applied in various industrial sectors where efficiency and flexibility of production are important.

Keywords: branch and bound method, NP-hard problems, genetic algorithm, genetic algorithm optimization, scheduling theory, genetic algorithm hybridization

For citation. Gorovykh II, Gorovykh SN. A Hybrid Approach to Solving NP-hard Production Problems: Combination of the Branch and Boundary Method with a Genetic Algorithm. *Young Researcher of Don.* 2024;9(1):10–19.

Введение. Научные исследования показали, что метод ветвей и границ является одним из наиболее эффективных алгоритмов для решения NP-сложных задач оптимизации. Однако существует несколько проблем, которые ограничивают его применение в реальных условиях. Во-первых, классический метод ветвей и границ

требует полного перебора всех возможных вариантов ветвления, то есть больших вычислительных ресурсов [1]. Это препятствует применению метода для задач большого масштаба. Во-вторых, метод ветвей и границ является детерминированным алгоритмом, который не учитывает случайных факторов или неопределенности, существующих в реальных производственных средах. Это может приводить к неоптимальным решениям или невозможности нахождения решения вообще.

Для преодоления этих ограничений авторы предлагают разработать гибридный подход, который объединяет метод ветвей и границ с генетическим алгоритмом. Генетический алгоритм позволяет использовать эволюционный подход и адаптивно настраивать параметры ветвления, учитывая зависимости и ограничения, имеющиеся в производственных процессах. Данная методология предлагает использовать генетический алгоритм для генерации и выбора оптимальных ветвлений в методе ветвей и границ, что позволяет сократить количество перебираемых вариантов и повысить производительность алгоритма.

Дополнительно в исследовании обращается внимание на применение предложенного гибридного метода для оптимизации планирования настроек производственного канала. Проведены эксперименты с использованием реальных данных производства для проверки эффективности и практической применимости предложенного подхода.

Ожидается, что разработанный гибридный метод ветвей и границ с генетическим алгоритмом будет иметь существенные преимущества по сравнению с традиционным методом ветвей и границ. Он будет способен эффективно решать сложные задачи оптимизации в производственных средах, обеспечивая повышение производительности и гибкости управления производственным процессом [2].

В дальнейшем авторы намерены представить подробное описание разработанной методологии, результаты экспериментов и их анализ, провести сравнение с другими существующими методами оптимизации, рассмотреть преимущества и ограничения данного подхода [3]. Они надеются, что это исследование будет иметь практическую ценность и поможет организациям достигнуть оптимального планирования и улучшения управления производственными процессами.

Основная часть. Описание проблемы. Пусть n — количество машин производственного канала и количество задач планирования. Для каждой машины (или задачи) M_i, $i \in \{1, ..., n\}$ известна дата выпуска R_i. Это минимальная продолжительность, необходимая для последнего шарикового подшипника предыдущей партии, чтобы перейти от M_i до MI. Также известен его хвост Q_i, минимальная продолжительность, необходимая для первого шарикового подшипника следующей партии, чтобы перейти от M_i до M_n. Когда машина M_i перезапускается, она не может оказать какого-либо влияния на производственную скорость, которая измеряется в конце производственной линии до блока времени Q_i. T_i и C_i обозначают начало и время завершения задачи настройки на машине M_i.

В производственной единице имеется λ операторов. Каждому оператору O_h, $h \in \{1, ..., \lambda\}$ в зависимости от собственного опыта требуется разное время для настройки машины, это время обозначается $p_{i,h}$. Если оператор O_h не умеет обращаться с машиной M_i, то без ограничения общности предположим $p_{i,h} = +\infty$. Кроме того, каждый оператор доступен только в течение временного интервала $[R_h D_h]$.

В данной статье исследуются методы оптимизации производственного процесса для последовательных каналов, где работа может возобновляться только после завершения процесса настройки всех машин. Один из ключевых показателей эффективности в таких условиях — это максимальное время выполнения задачи настройки $C_{max} = max_{i=1,...,n} C_i + q_i$, которое можно определить как максимальное значение между временем выполнения каждой задачи настройки C_i и временем, необходимым для подготовки следующей задачи q_i, где i — номер задачи.

Согласно классической классификации задач планирования, задача может быть определена как несвязанная задача многоцелевого планирования на параллельных машинах с заданными датами выпуска и хвостами. В данном случае ресурсы представляют собой операторов, а операции — задачи, которые необходимо выполнить на каждой машине. Формально эту задачу можно обозначить как $R,MPM\ |r_i, q_i|C_{max}$. На рис. 1 показан экземпляр задачи, состоящей из четырех машин и двух операторов.

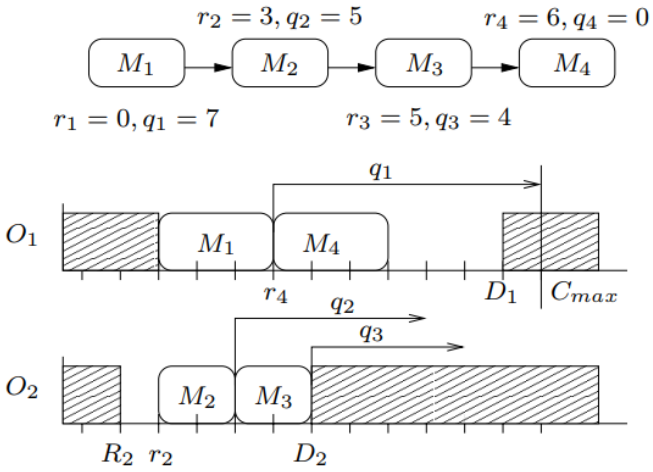
К тому, что было сказано ранее, можно добавить следующее: обозначения r_i и q_i в данной задаче можно интерпретировать как расстояния во времени от машины M_i до начала и конца производственного канала. Это означает, что для каждой машины чем больше расстояние до начала канала, тем ближе она к его концу. Таким образом, порядок неуменьшающихся значений r_i соответствует порядку неувеличивающихся значений q_i. Можно проиллюстрировать это утверждение (*утверждение 1*) на рис. 2.

Утверждение 1. В последовательном канале выпуска продукции для любого оператора i из множества $\{1, ..., n\}$ справедливо следующее: $r_i \le r_{i+1}$ *и* $q_i \ge q_{i+1}$.

Следствие 1. Оптимальным будет планирование задач в порядке неубывающей даты выпуска для каждого оператора.

Анализируя утверждение 1, можно легко вывести следствие 1, как было показано в работе Pessan и его соавторов [4]. Таким образом, проблему можно рассматривать как задачу назначения, где каждой задаче назначается определенный оператор, а затем определяется их порядок выполнения с учетом следствия 1. Такой подход позволяет оптимизировать заказы и время начала задач.

В контексте проблемы *R,MPM* r_i, $q_i|C_{max}$ это означает, что оптимальное планирование задач и определение временных интервалов начала и завершения является сложной задачей. Однако с использованием алгоритмических методов и оптимизационных подходов можно стремиться к поиску приближенного или эффективного решения в разумные сроки. Выводы исследования Гари и Джонсона дают понимание сложности данной проблемы и мотивируют исследователей и практиков разрабатывать новые методы и алгоритмы для ее эффективного решения [5].



$p_{i,h}$	O_1	O_2
M_1	3	2
M_2	2	2
M_3	∞	2
M_4	3	∞

	O_1	O_2
R_h	3	7
D_h	2	12

Рис. 1. Производство с двумя операторами и четырьмя машинами [4]

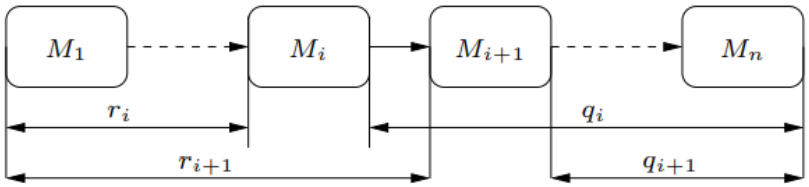


Рис. 2. Свойство на r_i и q_i [4]

Задача *R,MPM* $|r_i, q_i|C_{max}$ представляет собой интересную область исследования, которая не получила должного внимания в литературе, по сравнению с другими связанными проблемами расписаний, такими как *P* $|r_i, q_i|C_{max}$ и многоцелевые параллельные машины. Хотя работы Карлье (1987) и Гарби с Хауари (2002) и затрагивают аспекты этих проблем, они не исследовали данную конкретную задачу.

Метод ветвей и границ. В этом разделе кратко описывается метод ветвей и границ, представленный Pessan et. al. [4]. Метод ветвей и границ используется в гибридном методе. Он представляет собой эффективный подход к решению задачи, где требуется найти оптимальное решение в большом пространстве поиска. Основная идея метода заключается в том, чтобы разбить пространство поиска на подобласти и систематически исследовать их с целью определения оптимального решения.

Пространство поиска ассоциируется с деревом, где каждый узел дерева представляет собой частичное решение задачи (рис. 3). Начальным узлом является корень дерева, который соответствует начальной подобласти пространства поиска. Для каждого узла дерева определяются возможные варианты продолжения исследования. Эти варианты соответствуют различным подобластям пространства поиска, которые добавляются в стек, содержащий неизученные узлы.

На каждой итерации узел *(N)* извлекается из стека и вычисляется нижняя граница *lb(N)* для всех решений в соответствующем ему подпространстве. Затем эта нижняя граница сравнивается с лучшим известным решением *ub*, которое также называется верхней границей оптимального решения. Если *(N)* является конечным узлом и его критерий лучше верхней границы, верхняя граница обновляется. В противном случае, если нижняя граница больше верхней границы, то есть *lb(N)* > *ub*, узел отбрасывается (также утверждается, что узел обрезан),

а если он ниже, генерируются дочерние узлы и помещаются в стек. Процесс продолжается до тех пор, пока стек не станет пустым или не будет найдено оптимальное решение. При достижении конечного узла все переменные фиксируются и формируется окончательное решение задачи.

Важно отметить, что метод ветвей и границ может использоваться для решения различных задач, таких как задачи коммивояжера, раскраски графа и других комбинаторных задач. Этот метод является эффективным инструментом для поиска оптимальных решений в большом пространстве поиска и может быть адаптирован под различные требования исследуемой задачи.

На каждой итерации алгоритма метода ветвей и границ из стека извлекается узел *(N).* Затем для этого узла вычисляется нижняя граница *lb(N),* которая представляет собой оценку минимального значения всех возможных решений в соответствующем подпространстве. Далее полученная нижняя граница сравнивается с верхней границей *ub,* которая является лучшим известным решением, то есть оптимальным значением на текущий момент. Если узел *(N)* является конечным узлом и его критерий (нижняя граница) лучше, чем верхняя граница, то верхняя граница обновляется. Если нижняя граница больше верхней границы, *lb(N) > ub,* это означает, что для данного подпространства не существует перспективных решений, которые могли бы стать оптимальными. Этот узел, называемый неперспективным, отбрасывается (обрезается). Если же нижняя граница оказывается меньше верхней границы *lb(N) < ub,* то генерируются и добавляются в стек дочерние узлы, которые представляют решения для более узких областей поиска. Таким образом, алгоритм продолжает исследовать подпространства, которые потенциально могут содержать оптимальное решение.

Правило обрезки *(cut-off)* связано с использованием нижней границы для отсечения неперспективных узлов, которые не имеют значения для достижения оптимального решения. Правило вырезания *(pruning)* определяет процедуру, которая проверяет, есть ли в данном подпространстве перспективные решения с периодом разработки меньшим или равным заданному значению *D*. Если таких решений не обнаружено, то данный узел также может быть обрезан. Таким образом, использование правил обрезки и вырезания позволяет уменьшить количество рассматриваемых узлов дерева поиска и ускорить процесс поиска оптимального решения.

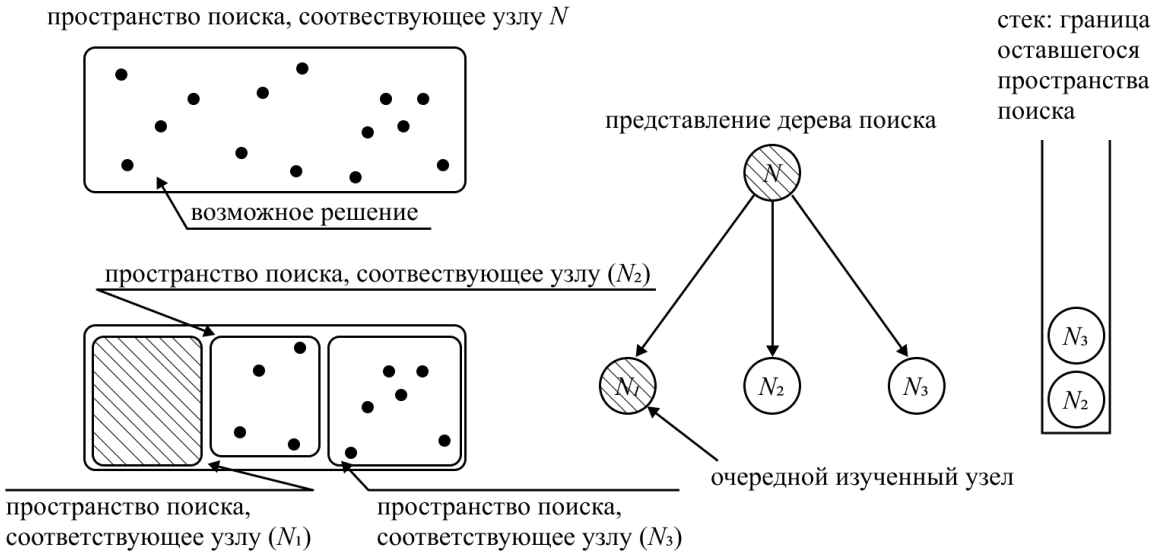


Рис. 3. Связь между пространством поиска, стеком и деревом [5]

Стратегия наилучшего начала подразумевает, что узлы с более низкой нижней границей будут рассматриваться первыми. Это позволяет преждевременно идентифицировать неперспективные узлы и отбросить их, что может существенно ускорить процесс поиска оптимального решения.

По результатам исследования 1 можно установить схему ветвления, где необходимо принять единственное решение — распределение задач между операторами. Более того, предполагается, что задачи уже упорядочены таким образом, что для любого $i \in \{1, ..., n-1\} \leq r_i + 1$. На каждом уровне (i) дерева поиска данная схема ветвления старается назначить задачу M_i+ каждому оператору, обладающему навыками, необходимыми для выполнения данной задачи. Это означает, что в дереве имеется n уровней и максимум λ ветвей на каждом узле, в зависимости от числа операторов, обладающих соответствующим навыком.

Максимальное значение. Для расчёта максимального значения используется простой жадный алгоритм, основанный на правиле «Наименьшее время окончания» (ECT). Этот алгоритм назначает задачи операторам, которые смогут выполнить их раньше всего. При экспериментальных исследованиях (Pessan et. al.) было обнаружено, что этот верхний предел является основной проблемой метода, так как он слишком отличается

от оптимального значения для уменьшения количества узлов. Кроме того, решения, полученные с использованием ECT, могут быть невозможными в отношении интервала $[R_h D_h]$.

Правило вырезания — это интересный подход, основанный на ослаблении ограничений приоритетов выполнения задач. В его основе лежит идея о том, что задача может быть прервана и перезапущена позже на том же операторе или на другом. Чтобы определить сроки выполнения каждой задачи, используется формула $\tilde{d}_l = ub - 1 - q_i$, где ub — верхняя граница временного окна, а q_i — время начала задачи. Польза этого правила заключается в том, что оно дает возможность проводить проверку выполнения всех задач в рамках разрешенных временных окон, которые определены как $\left[r_i, \tilde{d}_l\right]$. Если такая проверка показывает, что выполнение всех задач невозможно, то можно применить правило обрезания узла. Интересно, что эту проверку можно выполнить полиномиально, с использованием линейной программы, как было продемонстрировано Лоулером и Лабетулем еще в 1978 году.

Таким образом, применение правила вырезания и его полиномиальной проверки позволяет эффективно управлять временными ограничениями и улучшать выполнение задач. Оно расширяет возможности управления процессом и обеспечивает более гибкое выполнение задач.

Гибридный метод. Гибридный метод объединяет преимущества метода ветвей и границ с использованием условия вырезания. Метод ветвей и границ является эффективным способом перебрать все допустимые решения. Однако при больших пространствах поиска это может быть крайне ресурсоемким процессом. Важной частью этого гибридного метода является улучшение верхней границы. Чем ближе верхняя граница к оптимальному значению, тем эффективнее отсечение узлов и уменьшение пространства поиска. Были разработаны эвристики и оптимизированные алгоритмы, чтобы как можно быстрее находить более точную верхнюю границу.

Важно отметить, что в случаях, когда поиск останавливается до достижения оптимального решения, метод стремится предоставить оператору хорошее приближенное решение. Вместо продолжения поиска оптимального решения оператору предлагается лучшее найденное решение на данный момент. Это полезно, особенно когда оператор ограничен во времени или ресурсах, а решение принять требуется как можно скорее.

Рассмотрим возможность оптимизировать метод ветвей и границ с помощью генетического алгоритма. Генетический алгоритм будет применяться для эффективного вычисления верхней границы проблемы. Этот алгоритм может быть использован как на стадии корневого узла, так и на любой другой стадии поиска с целью нахождения лучшей верхней границы. Идея заключается в использовании генетического процесса для улучшения наилучшего известного решения путем исследования оставшегося пространства поиска, то есть области, границы которой все еще находятся в стеке ветвей и границ. При этом ненужная часть уже исследованного пространства будет исключена.

Существует два основных способа изменить пространство поиска. Во-первых, при создании дочерних узлов в процессе ветвления и границ пространство поиска родительского узла делится на объединенные подпространства для новых узлов. Во-вторых, при обрезании узлов в процессе ветвления и границ часть пространства поиска удаляется. В сущности, генетический алгоритм должен быть реализован таким образом, чтобы он искал только в неисследованных областях, а удобным способом для этого является использование узлов, которые все еще находятся в стеке ветвей и границ.

Гибридный точный генетический алгоритм. Генетический алгоритм — это метод решения оптимизационных задач, основанный на принципах биологической эволюции. Он был предложен Холландом в 1975 году. В основе генетического алгоритма лежит понятие популяции индивидов. Каждый индивид представляет собой потенциальное решение задачи и представлен в виде хромосомы, состоящей из генов. Гены содержат информацию о параметрах или переменных, на основе которых определяется решение задачи.

Алгоритм выполняет следующие шаги в каждой итерации:

1. Инициализация популяции: создается начальная популяция случайных индивидов.
2. Оценка приспособленности: каждому индивиду присваивается значение функции приспособленности, которое является критерием качества решения.
3. Отбор: индивиды с более высокой приспособленностью имеют больше шансов быть выбранными для попадания в следующее поколение.
4. Кроссовер: случайным образом выбираются два родителя из популяции, и их гены смешиваются, создавая новых потомков.
5. Мутация: некоторые гены в потомках могут быть изменены случайным образом, чтобы внести разнообразие в популяцию.
6. Замена поколения: лучшие индивиды сохраняются, а остальные замещаются новыми потомками.

7. Повторение: шаги 2–6 повторяются до достижения условия останова, например, максимального числа итераций или достижения определенного уровня приспособленности.

Главная идея генетического алгоритма заключается в эволюции популяции индивидов и постепенном улучшении качества решения. Путем сочетания перекрестных операций и случайных мутаций алгоритм исследует пространство возможных решений и находит оптимальное решение задачи оптимизации. Функция кодирования: как было указано ранее, первое следствие позволяет свести нашу задачу к проблеме назначения операторов. Следовательно, кодирующая функция генетического алгоритма должна состоять только из операторов, которые присваивают задачам определенные значения. В этом контексте функция декодирования генетического алгоритма должна осуществлять упорядочивание задач в неубывающем порядке r_i, а затем определять время начала выполнения каждой задачи.

Выбранная функция кодирования оперирует с тремя основными компонентами, которые формируют отдельные связи: узел, уровень узла и массив присваиваний. Структура узла содержит частичное решение и его позицию в дереве поиска: узел на уровне k содержит k заданий. Таким образом, первая часть индивида, а именно узел личности, определяет и обозначает начальные присваивания. В свою очередь, остальные присваивания должны быть закодированы в третьей части индивида. Данная методика позволяет обеспечить более значимую и уникальную форму представления данных.

Индивидуум, представленный в таблице 1, использует узел 7 на уровне 2 дерева поиска. Первые два задания берутся из этого узла. Узел содержит только частичное решение с двумя заданиями, поэтому для завершения оставшейся части индивидуума требуется еще 4 задания с уровня n узлов, равного $n - 2 = 4$.

Таблица 1

Пример закодированного решения задачи с n = 6 и m = 4

Узел	Уровень узла	Назначение оставшихся задач
7	2	1/3/4/4

Оператор кроссовера, который используется здесь, это классический одноточечный кроссовер. Он порождает двух потомков от двух родителей. Сначала случайным образом выбирается число p от 0 до n. Затем первый потомок генерируется путем копирования p присваиваний от первого родителя и $n - p$ присваиваний от второго родителя. Узел потомка соответствует узлу первого родителя. Второй потомок создается путем обмена ролями родителей.

Преимущество этого оператора кроссовера заключается в том, что дочерний элемент всегда содержит существующие узлы стека и действительные назначения и, следовательно, принадлежит оставшемуся пространству поиска. Границей этого пространства являются узлы стека ветвей и границы дерева.

На примере на рис.5 p установлено равным 3. Итак, от первого индивидуума к первому потомку копируются 3 задания: 2 задания узла и одно дополнительное задание. Затем от второго индивидуума берутся другие назначения: одно из узла 21, а остальные — из 3 поля индивидуума.

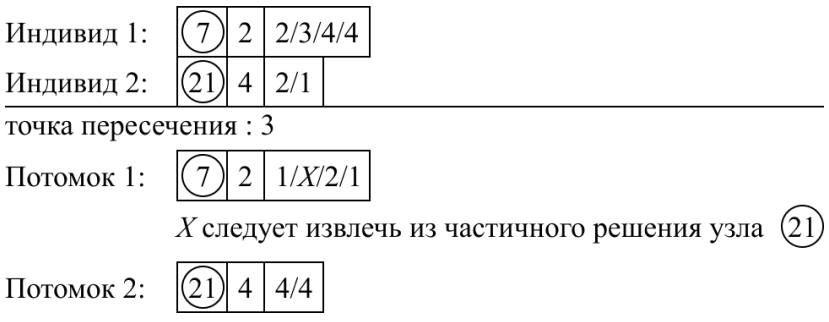


Рис. 5. Пример кроссовера

Оператор мутации: в методе мутации используются два оператора. Первый оператор случайным образом изменяет ген внутри третьего поля у особи, которое содержит задания для завершения частичного решения узла. Второй оператор мутации случайным образом переключает узел на новый, который находится в стеке. Если новый узел находится на более низком уровне, то он извлекает недостающие задания из старого узла.

В примере, представленном на рис. 6, узел 7, находящийся на уровне 2, заменяется на узел 21, расположенный на уровне 4. Первые четыре задания мутировавшего индивидуума берутся из узла 21, а оставшиеся два задания сохраняются от исходного индивидуума.

Рис. 6. Пример второго оператора мутации

Вероятность *pm1* мутации особи должна быть установлена на высокое значение в основном из-за выбранной функции кодирования и оператора кроссинговера. Это обеспечивает введение новых присваиваний в популяцию при отсутствии мутаций. Однако для быстрого исследования подпространства в данном гибридном методе такая мутация необходима.

Если имеется мутация, то вероятность использования второго оператора pm2 должна быть низкой. Поскольку этот оператор изменяет множество назначений у индивидуума, его слишком частое применение может привести к излишней степени случайности.

Оператор синхронизации является специфичным для данного гибридного метода. Он используется для проверки того, что генетический алгоритм исследует только неизведанную область поиска, ограниченную границей стека. Из-за значительного времени выполнения этого оператора он не должен вызываться на каждой итерации. Слишком частое применение оператора может затруднить корректное развитие генетического алгоритма, так как он может делать недействительными решения, как только они появляются. Поэтому рекомендуется использовать этот оператор каждые *maxIt (maxIt = 2000)* итераций генетического алгоритма. Оператор синхронизации просто проверяет, что у всех индивидуумов генетического алгоритма есть узел, который все еще находится в стеке. Если обнаруживается узел, который больше не находится в стеке, вызывается второй оператор мутации, который изменяет узел индивидуума.

Если улучшение наилучшего решения при использовании генетического алгоритма не происходит в течение значительного времени и если метод выполняется на однопроцессорной системе, то имеется возможность временно приостановить работу генетического алгоритма, чтобы уделить больше процессорного времени методу ветвей и границ. Это объясняется тем, что во многих случаях требуется время, чтобы доказать, что было найдено оптимальное решение и дальнейшее улучшение уже найденного решения может быть неэффективным. Кроме того, оператор синхронизации предоставляет список решений, которые имеют оценку, равную верхней границе метода ветвей и границ. Затем метод ветвей и границ может быстро исследовать соответствующие узлы. Это имеет два преимущества:

– удаление этих решений из стека и запуск генетического алгоритма для поиска в других областях пространства поиска;

– генерирование дочерних узлов для всех узлов, находящихся на пути к этим решениям, и добавление в стек некоторых соседних узлов лучших решений генетического алгоритма. Это позволяет потенциально достичь как лучшего результата методом ветвей и границ, так и решения при помощи генетического алгоритма.

Стратегия исследования: классический метод ветвей и границ обеспечивает быстрый поиск допустимых решений и ограниченный размер стека. Однако в данном случае такой подход может быть не наилучшим, так как все узлы стека находятся в одной области пространства поиска, что не способствует эффективной работе генетического алгоритма. В связи с этим была реализована другая стратегия, называемая «наилучшая первая». Она заключается в выборе узла с наименьшей нижней границей, а в случае равных нижних границ выбирается самый глубокий узел. Основное преимущество такой стратегии заключается в формировании стека достаточно большого размера, содержащего узлы, распределенные по всему пространству поиска. Это дает генетическому алгоритму больше возможностей для эффективной работы. Более того, такая стратегия может улучшить нижнюю границу общей задачи, что является полезной информацией для принятия решений оператором, который определяет, стоит ли продолжать долгий поиск оптимального решения. Идея данной стратегии заключается в использовании метода ветвей и границ для отсечения максимального количества узлов и, наконец, в доказательстве оптимальности найденного решения, в то время как генетический алгоритм занимается поиском хороших решений. Стратегия изображена на рис. 7.

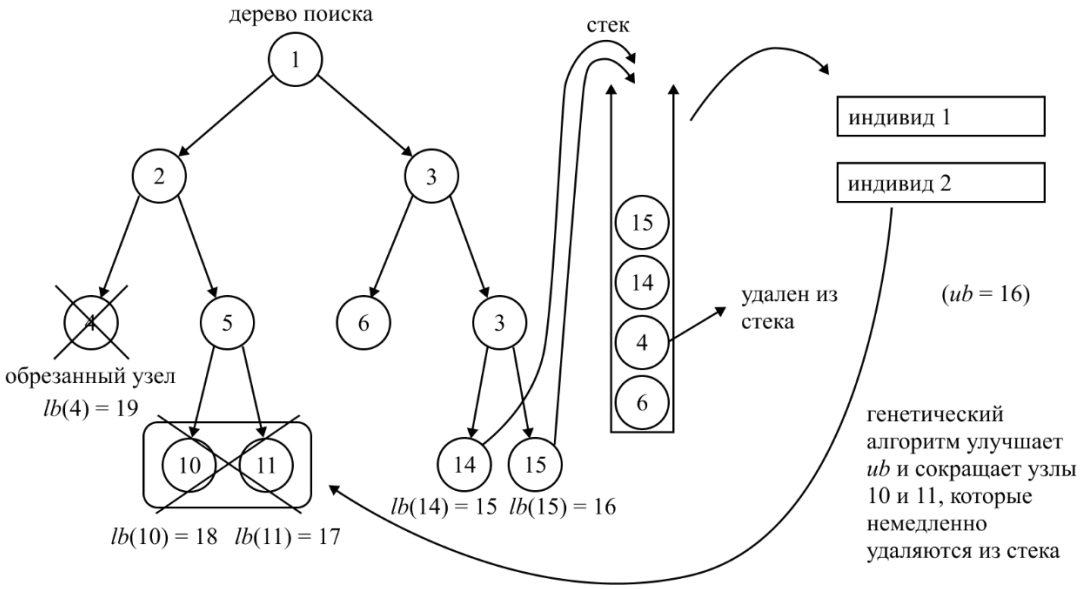


Рис. 7. Синхронизация стека с генетическим алгоритмом и влияние генетического алгоритма на метод ветвей и границ [6]

Обратите внимание, что есть возможность вычислить нижнюю границу при отсутствии явной границы. Это можно сделать, используя условие разделения. Представим D^* как наименьшее значение D, при котором условие разделения не выполняется. Тогда значение $D^* + 1$ будет допустимой нижней границей. В узлах N вычисляются различные значения D, начиная с нижней границы родительских узлов N, чтобы определить допустимую нижнюю границу. В корневом узле lb вычисляется с помощью бинарного поиска по D. Кроме того, при вызове оператора синхронизации все особи, у которых критерий соответствует наилучшему найденному решению на данный момент, отправляются в метод обработки приоритетов узлов. Затем метод ветвей и границ немедленно исследует эти узлы. Таким образом, они удаляются из стека, и генетический алгоритм будет вынужден искать решения, отличные от этих лучших при последующей синхронизации.

Результаты экспериментов. Результаты экспериментов, проведенных авторами, получились весьма обнадеживающими. Был реализован метод с использованием языка программирования Java, он был протестирован на многопроцессорной машине с процессором, работающим на частоте 3,7 МГц и 8 ГБ оперативной памяти. При этом два алгоритма работают в собственных потоках, что позволяет программе эффективно использовать второй процессор на многопроцессорной или многоядерной машине.

После предварительных тестов стало ясно, что определенные параметры приводят к хорошим результатам. Была использована численность популяции в 500 особей, вероятность мутации pm = 30 %, вероятность мутации второго типа $pm\ 2$ = 5 % и максимальное число итераций $maxIt$ = 2000. Также была ограничена продолжительность метода до 10 минут. Для проведения тестов было сгенерировано 2000 экземпляров с разным количеством задач (от 10 до 45) и операторов (от 2 до 10). При генерации создаются экземпляры, которые были бы схожи с промышленными вариантами в терминах навыков и значений. В типичном промышленном случае количество задач составляет от 30 до 40. В таблице результатов используются следующие обозначения: *bb* — это метод ветвей и границ только с ветвлением и привязкой, *df* — это гибридный метод с поиском в глубину, а *bf* — это гибридный метод с поиском в первую очередь. Одинокий запуск метода ветвей и границ использовал поиск в глубину, так как использование наилучшего первого поиска приводит к увеличению размера стека из-за неоптимальной верхней границы.

По данным таблицы 1 можно заметить, что гибридный метод с поиском в глубину обычно не уступает или даже превосходит метод ветвей и границ, несмотря на то, что в гибридном методе исследуется меньшее количество узлов, так как оба метода используют процессор в совместной работе. Среди двух гибридных методов лучшим оказывается первый метод для больших экземпляров с высокими различиями, начиная с n = 30. Это соответствует размеру экземпляров, с которыми была произведена работа в промышленном примере.

Данные таблиц 2 и 3 показывают разрыв в процентном соотношении между нижней границей всех узлов, которые остаются в стеке, и верхней границей. Это говорит о том, что гибридный метод с поиском в глубину редко улучшает этот разрыв, но лучший первый метод существенно его улучшает. Это объясняется двумя факторами. Первый фактор заключается в том, что разнообразие содержимого стека дает генетическому алгоритму больше возможностей для улучшения своего лучшего решения, чем локальный поиск, выполняемый

с помощью поиска в глубину. Второй фактор заключается в том, что при использовании лучшего первого метода нижняя граница оставшихся узлов естественным образом улучшается со временем.

Таблица 2

Процент случаев, решенных до оптимального [6]

m	*n*=10			*n*=15			*n*=20			*n*=25			*n*=30			*n*=35			*n*=40			*n*=45		
	bb	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*
2	100	100	100	100	100	100	100	100	100	100	100	100	100	100	97	97	97	97	97	97	97	83	83	83
3	100	100	100	100	100	100	100	100	100	100	100	97	97	100	97	80	87	93	70	70	77	50	47	77
4	100	100	100	100	100	100	100	100	97	80	87	90	73	73	80	50	50	57	43	43	47	27	33	50
5	100	100	100	100	100	100	83	83	83	77	80	77	47	47	63	40	43	57	23	20	27	3	7	37
6	100	100	100	100	100	100	90	93	93	60	63	67	23	23	37	20	23	27	3	3	7	6	7	7
7	100	100	100	97	100	97	60	70	60	60	60	57	13	13	20	3	13	20	3	7	3	0	7	7
8	100	100	100	93	93	83	47	50	60	33	33	30	17	20	23	13	13	13	3	7	3	6	7	7
9	100	100	100	90	93	90	53	53	37	23	27	30	12	3	0	3	7	7	0	3	7	3	3	3
10	100	100	100	90	93	90	13	17	13	10	13	13	7	7	3	3	3	0	0	3	3	3	3	3

Таблица 3

Средний разрыв между нижней и верхней границей [6]

m	*n*=10			*n*=15			*n*=20			*n*=25			*n*=30			*n*=35			*n*=40			*n*=45		
	bb	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*	*bb*	*df*	*bf*
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,03	0,01	0,01	0,01	0,05	0,05	0,05	0,3	0,3	0,3
3	0	0	0	0	0	0	0	0	0	0	0	0,01	0,2	0,1	0,02	0,5	0,5	0,09	0,7	0,7	0,4	1,7	1,7	0,2
4	0	0	0	0	0	0	0	0	0,02	0,6	0,6	0,1	0,9	0,9	0,2	1,9	1,9	0,6	2,7	2,7	0,5	2,9	2,9	0,6
5	0	0	0	0	0	0	0,9	0,9	0,5	0,9	0,8	0,3	2,7	2,7	0,4	3,1	3,1	0,6	3,0	3,0	1,0	5,5	5,5	1,6
6	0	0	0	0	0	0	0,4	0,4	0,1	2,0	2,0	0,7	3,7	3,7	1,1	4,7	4,7	1,4	7,7	7,7	2,3	6,3	6,3	2,1
7	0	0	0	3,0	3,0	0,03	0,6	5,6	1,6	3,0	3,0	2,0	6,0	6,0	2,9	6,4	6,4	2,1	7,0	7,0	3,4	8,0	8,0	3,7
8	0	0	0	5,0	5,0	2,0	11,0	11,0	6,3	5,1	5,1	2,5	10,0	10,0	3,5	7,9	7,9	3,3	8,0	8,0	4,2	9,1	9,2	4,2
9	0	0	0	4,0	3,0	1,5	13,0	13,0	8,2	15,0	15,0	11,0	9,4	9,4	4,8	8,8	8,8	5,2	9,6	9,6	4,5	10,5	10,5	5,5
10	0	0	0	4,0	3,0	1,5	25,0	25,0	13,0	28,0	28,0	14,0	14,0	14,0	5,9	9,0	9,0	5,5	11,0	11,0	5,3	10,4	10,4	6,1

Заключение. Авторами разработан инновационный подход, который объединяет генетический алгоритм и метод ветвей и границ с целью повышения эффективности обоих методов. Генетический алгоритм, по сути, направлен на сокращение пространства поиска, а метод ветвей и границ является мощным инструментом для этой цели. Данный подход заключается в том, чтобы использовать генетический алгоритм для улучшения верхней границы, что, в свою очередь, сокращает количество узлов, которые нужно обрабатывать методом ветвей и границ.

Проведенные эксперименты показали многообещающие результаты данного подхода, особенно при использовании лучшего первого поиска. Кроме того, стоит отметить, что эти тесты проводились на шестипроцессорной машине, что значительно повышает потенциал данного подхода в случае использования многоядерных процессоров.

Использование этой комбинации методов позволяет добиться более эффективного и быстрого поиска оптимального решения для различных задач. Данный подход является перспективным и внесет значительные улучшения в область оптимизации и поиска решений.

Список литературы

1. Basseur M., Lemesre J., Dhaenens C., Talbi E.G. Cooperation between branch and bound and evolutionary approaches to solve a bi-objective flow shop problem. In: *Experimental and Efficient Algorithms. Lecture Notes in Computer Science*. Berlin: Springer, Heidelberg; 2004. P. 72–86. https://doi.org/10.1007/978-3-540-24838-5_6

2. Carlier J. Scheduling jobs with release dates and tails on identical parallel machines to minimize the makespan. *European Journal of Operational Research*. 1987;29(3):298–306. https://doi.org/10.1016/0377-2217(87)90243-8

3. Cotta C., Aldana J.F., Nebro A.J., Troya J.M. Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP. In: *Artificial Neural Nets and Genetic Algorithms. Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer; 1995. P. 277–280. https://doi.org/10.1007/978-3-7091-7535-4_73

4. Pessan C., Néron E., Bellenguez O. Modélisation et planification des opérations de réglage de machines lors de changements de série. In: *6ème Conférence Francophone de Modélisation et Simulation Modélisation, Optimisation et Simulation des Systèmes: Défis et Opportunités.* 2006. P. 1545–1554.

5. Garey M.R., Johnson D.S. Strong NP-Completeness results: motivations, examples, and implications. *Journal of the ACM (JACM).* 1978;25(3):499–508.

6. Jouglet A., Sevaux M., Oguz C. Flowshop hybride: de nouvelles perspectives en mˆelant algorithme g´en´etique et propagation de contraintes. In: *ROADEF 2005, Congrés de la Société Française en Recherche Opérationnelle*. France, Tours; (2005). P. 41.

7. Anis Gharbi, Mohamed Haouari. Minimizing makespan on parallel machines subject to release dates and delivery times. *Journal of Scheduling.* 2002;5(4):329–355. https://doi.org/10.1002/jos.103

Об авторах:

Иван Иванович Горовых, аспирант кафедры программного обеспечения вычислительной техники и автоматизированных систем Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), vansk1997@gmail.com

Светлана Николаевна Горовых, аспирант кафедры программного обеспечения вычислительной техники и автоматизированных систем Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1).

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the Authors:

Ivan I. Gorovykh, Postgraduate student of the Computer Engineering and Automated Systems Software Department, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, RF), vansk1997@gmail.com

Svetlana N. Gorovykh, Postgraduate student of the Computer Engineering and Automated Systems Software Department, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, RF).

Conflict of interest statement: the authors do not have any conflict of interest.

All authors have read and approved the final manuscript.