Раздел 2
МОДЕЛИ, СИСТЕМЫ, СЕТИ В ТЕХНИКЕ

Section 2
MODELS, SYSTEMS, NETWORKS IN THE TECHNIQUE

УДК 004.023
doi: 10.21685/2227-8486-2025-3-7

РАЗДЕЛЕНИЕ ОЧЕРЕДЕЙ КАК МЕТОД ОПТИМИЗАЦИИ АЛГОРИТМА МУРАВЬИНОЙ КОЛОНИИ ПРИ ПОСТРОЕНИИ РАСПИСАНИЯ МНОГОСТАНОЧНОГО ЦЕХА РАЗОВОГО ПРОИЗВОДСТВА СО СБОРОЧНЫМИ ОПЕРАЦИЯМИ

М. А. Иванов

НефтемашСистема, Саратов, Россия
mikeaivanov@gmail.com

Аннотация. *Актуальность и цели.* Рассматривается применение алгоритма муравьиной колонии для решения задачи составления расписания для цеха единичного производства со сборочными операциями (assembly job shop). Он дает хорошие результаты, однако из-за высокой вычислительной сложности слабо подходит для решения реальных практических задач с размерностью в сотни и тысячи операций. *Материалы и методы*. Предлагается метод существенного повышения вычислительной эффективности данного алгоритма применительно к многостаночной задаче – наиболее часто встречающемуся на практике случаю. Он состоит в разделении единой очереди работ на отдельные очереди по типам используемых станков и применении алгоритма муравьиной колонии к каждой из них по отдельности. *Результаты*. Показывается, что на крупных (тысячи операций) задачах временные затраты сокращаются в 10 и более раз без потери качества находимых решений. *Выводы*. Предложенная модификация позволяет существенно улучшить производительность алгоритма муравьиной колонии при практическом применении для решения производственных задач машиностроительных предприятий.

Ключевые слова: расписание производства, цех единичного производства со сборками, метод муравьиной колонии, метаэвристические методы, многоуровневые сборки

Благодарности: автор благодарит за ценные замечания к.т.н. П. В. Матренина.

Для цитирования: Иванов М. А. Разделение очередей как метод оптимизации алгоритма муравьиной колонии при построении расписания многостаночного цеха разового производства со сборочными операциями // Модели, системы, сети в экономике, технике, природе и обществе. 2025. № 3. С. 89–101. doi: 10.21685/2227-8486-2025-3-7

QUEUE SEPARATION AS A METHOD OF OPTIMIZATION OF ANT COLONY ALGORITHM FOR MULTI-MACHINE ASSEMBLY JOB SHOP SCHEDULING

M.A. Ivanov

NeftemashSistema, Saratov, Russia
mikeaivanov@gmail.com

Abstract. *Background*. Ant Colony Algorithm is an effective method of solving assembly job shop problem. But its computational complexity makes it impractical for real-life tasks of hundreds and thousands of operations. *Materials and methods*. This article describes a method to significantly increase computational effectiveness of the algorithm for multi-machine problem – the most important case for practical use. The proposed method uses separation of the task queue into a number of subqueues for each type of the processing machines and applying Ant Colony Optimization to each of them separately. *Results*. It is shown that for large-scale problems (thousands of operations) runtime decreases 10 times or more without losing the quality of solutions. *Conclusions*. This modification significantly increases ACO algorithm speed when applied to practical tasks of machine manufacturing plants.

Keywords: assembly job shop, assembly job shop scheduling, ant colony optimization, metaheuristics, multi-level assemblies

Acknowledgements: the author thanks P. V. Matrenin, candidate of technical sciences, for his valuable comments.

For citation: Ivanov M.A. Queue separation as a method of optimization of ant colony algorithm for multi-machine assembly job shop scheduling. *Modeli, sistemy, seti v ekonomike, tekhnike, prirode i obshchestve = Models, systems, networks in economics, technology, nature and society*. 2025;(3):89–101. (In Russ.). doi: 10.21685/2227-8486-2025-3-7

Введение

Сокращение сроков производства продукции – один из важных факторов конкурентоспособности предприятия. Одним из способов достижения этого является оптимизация порядка производства деталей и узлов. Задача составления расписания производства является NP-сложной, и однозначного алгоритма ее оптимального решения не существует. Ее двумя основными подвидами являются задачи о цехе разового (job shop) и поточного производства (flow shop). В первом случае на последовательность операций никакого ограничения не накладывается, во втором считается, что порядок прохождения станков для всех деталей одинаков (некоторые станки при этом могут быть пропущены). Более сложным вариантом данных задач являются задачи со сборочной стадией (assembly job / flow shop). В таких задачах предполагается, что после изготовления детали поступают на сборочную операцию, в рамках которой объединяются в узлы; те, в свою очередь, могут служить комплектующими для узлов более высокого уровня. Мы рассматриваем вариант, когда количество уровней сборок заранее неизвестно и неограниченно. Обычно считается,

что сборка может начаться только после того, как готовы все детали; на практике это не всегда так, но далее мы будем придерживаться данного допущения.

Правила диспетчирования были первым, вычислительно наиболее простым методом решения задачи составления расписания. Такие правила вычисляют приоритеты выполнения операций, исходя из их продолжительности и общего числа операций, нужных для изготовления детали. Изучение подобных подходов провели, например, Фрай (с соавт.) [1] и Филипум (с соавт.) [2], рассмотрев некоторые правила диспетчирования для задачи минимизации времени изготовления и совокупного запаздывания. Доктор (с соавт.) [3] разработал эвристический алгоритм, оптимизировавший расписание для изготовления изделий с не более чем тремя уровнями сборочных единиц. МакКой и Эгбелу [4] и Парк и Ким [5] разработали эвристические процедуры решения задачи оптимизации изготовления двухуровневого изделия. Мун-Вон и Ким [6] разработали модель на основе частично-целочисленного линейного программирования и алгоритм по методу ветвей и границ, находивший оптимальные решения случайно сгенерированных двухуровневых задач умеренного размера. Риджа и Раджендран [7] предложили концепцию синхронизации операций по дате (OSD): расписание должно составляться так, чтобы изготовление деталей, входящих в одну сборку, завершалось как можно более синхронно. Цао (с соавт.) предложил метод составления расписания инвертированием дерева изделия [8].

Важным достоинством правил диспетчирования является их низкая вычислительная сложность. Однако, как показали Омкумар и Шахабудин [9], все они уступают в качестве получаемых решений методу муравьиной колонии. Для решения задачи составления расписания используют и другие итеративные методы, например генетические алгоритмы, метод симулирования отжига и т.п., но они не всегда хорошо справляются с ограничениями, появляющимися, если мы рассматриваем многоуровневую сборочную структуру. Важным достоинством метода муравьиной колонии является то, что порождаемые им решения всегда являются допустимыми, так как учитывают структуру сборочных единиц независимо от ее сложности. В случае задач существенной размерности использование эвристической функции позволяет сузить область поиска, исключая из рассмотрения неэффективные направления. Наконец, при необходимости учесть какие-то дополнительные условия производственных задач, это достаточно легко сделать, не меняя принципиальную структуру алгоритма. Примерами таких условий могут быть отказ от требования доступности любых деталей для изготовления с момента $T = 0$ или добавление возможности выполнять детали не на единственном станке, а на любом из нескольких идентичных. Для их выполнения достаточно изменить функцию, помещающую операции в очередь работ станка.

Недостатком этого итеративного метода являются существенно большая вычислительная сложность и, соответственно, время работы. Для практических задач, размерность которых может в сотни раз превосходить классические тестовые примеры, эта проблема остается актуальной.

Метод муравьиной колонии имитирует поведение ищущих источник пищи муравьев. Муравей оставляет за собой постепенно испаряющийся феромонный след. Соответственно, если муравей нашел источник пищи и принес ее в муравейник, то оставленный им след будет тем сильнее, чем меньше времени потребовалось на дорогу. Если другой муравей обнаружит такой след,

то вероятность того, что он пойдет по нему, тем выше, чем сильнее след. Таким образом, постепенно все муравьи начинают пользоваться самым коротким маршрутом к источнику пищи. Алгоритм, подражающий такому поведению, был предложен Марко Дориго [10]. Агенты («муравьи») совершают многочисленные последовательные попытки построения возможных решений. При этом они используют случайный выбор, основанный одновременно на эвристике и результатах, полученных в предыдущих попытках.

Современные исследования рассматриваемой задачи чаще всего ориентированы на улучшение результатов работы алгоритма посредством ускорения сходимости и предотвращения застревания в локальных минимумах. В исследовании [11] сравниваются несколько классических эвристик, а в работе [12] метод муравьиной колонии комбинируется с эвристикой скользящего узкого места. В статье [13] предлагается механизм повышения разнообразия начальных решений. В работе [14] используются множественные колонии и модифицируется механизм обновления феромонного следа. В исследовании [15] использование элитистской модификации алгоритма сочетается с механизмом диффузии феромона.

В ряде случаев алгоритм муравьиной колонии используется в сочетании с другими методами, например генетическими алгоритмами, как в работе [16].

Однако в случае больших задач из опыта реального производства (сотни и тысячи деталей) в силу сложности $O(N^2)$ метод муравьиной колонии становится непрактичным. В настоящей работе показано, что можно существенно (в самых крупных проанализированных задачах – в 10 и более раз) улучшить время работы алгоритма, рассматривая входящую очередь заданий не в целом, а сразу разделив ее по типам работ, при этом на крупных задачах полученный результат также улучшается.

Допущения

Кратко опишем допущения, которые мы делаем в данном исследовании:

1. Операции выполняются без прерываний (невозможно заменить задание по выполнению операции с Q деталями на два задания с Q_1 и Q_2 деталей, $Q = Q_1 + Q_2$).

Это стандартное допущение. Предполагается, что разделение деталей на партии, если оно нужно, уже проведено ранее и учтено в нормах времени на операцию. Кроме того, это связано с таким важным аспектом технологичности, как использование групповых заготовок.

Вполне очевидно, что разделение операций на партии никак не влияет на алгоритм в целом: просто вместо одной операции появляются несколько для одного и того же станка, которые алгоритм будет обрабатывать тем же способом.

2. Все станки доступны непрерывно начиная с момента $T = 0$ и могут выполнять не более одной задачи одновременно.

Это также стандартное допущение. Очевидно, что если какой-то станок недоступен в некий промежуток времени, то мы можем директивно добавить в расписание на это время задачу «Простой». Если станок может выполнять две задачи одновременно, то мы можем учитывать его как два независимых станка.

3. Каждый из станков предназначен для выполнения операций только одного типа.

Это также стандартное допущение, но содержащее повод для дальнейшего исследования. Многофункциональные обрабатывающие центры могут заменять различные станки, но, во-первых, таких аппаратов меньше, чем станков, предназначенных для выполнения одного типа операций, а во-вторых, это не влияет на алгоритм в целом. Если необходимо рассматривать многофункциональные станки, то предложенный алгоритм может выбирать либо специфический, либо многофункциональный станок (возможно, благодаря добавлению особой эвристики; можно предположить, что использование специфического станка, как правило, будет предпочтительным).

4. Каждому типу операций могут соответствовать несколько идентичных станков.

Как правило, такой вариант не рассматривается, но он соответствует реалиям работы предприятия, данные которого были использованы в качестве модельных примеров. Некоторые виды операций встречаются гораздо чаще других, и без рассмотрения множественных идентичных станков реальная производственная задача не имеет смысла: длительность расписания в целом будет зависеть почти исключительно от суммарной длительности всех операций наиболее популярного типа, а порядок выполнения операций других типов не будет иметь существенного значения.

5. Мы не учитываем время на транспортировку деталей между станками и доступность нужных для этого средств.

Очевидно, что при необходимости в технологию производства можно добавить транспортировочные операции с теми или иными требованиями (возможно, динамически меняющимися) к используемому для транспортировки оборудованию. В реалиях рассматриваемого предприятия этого не требуется (партии относительно малы, а средства транспортировки никогда не являются узким местом). В целом интеграция алгоритмов управления расписанием производства и автоматической транспортировки является темой отдельных исследований [17].

6. Не рассматриваются требования по неразрывному выполнению некоторых последовательных операций.

Мы не рассматриваем данное ограничение, чтобы не затруднять восприятие основной идеи исследования. Очевидно, что при необходимости проблема решается незначительной модификацией: если следующая операция должна выполняться неразрывно, то она сразу же будет помещаться в очередь работ рассматриваемого станка.

7. Сборочные работы не делятся по требованиям к квалификации; считается, что все они выполняются одной группой рабочих (т.е. у нас есть одна очередь для всех сборочных работ).

Очевидно, что данное ограничение также не влияет на алгоритм в целом и связано только с доступными тестовыми данными, в которых нет разделения сборочных операций по категориям. Если бы такое разделение было, то это добавило бы еще один или несколько видов операций и «станков» – групп обученных их выполнению сборщиков.

Таким образом, принятые для простоты изложения основной идеи допущения не уменьшают применимости алгоритма в целом, а исключение этих ограничений требует лишь незначительных модификаций базового алгоритма (и получения соответствующих исходных данных).

Метод муравьиной колонии

Используя данный метод, мы предпринимаем многочисленные попытки построить допустимое расписание производства. Эти попытки объединяются в поколения: в рамках каждого из них все попытки используют один набор данных о результатах предыдущих поколений («таблицу следов»), который обновляется перед переходом к следующему поколению.

На каждом шаге мы выбираем, какую из доступных операций поместить в расписание, пользуясь формулой

$$P_{ij}^{k} = \frac{f_{ij}^{\alpha} \cdot e_{ij}^{\beta}}{\sum_{j \in N_i^k} f_{ij}^{\alpha} \cdot e_{ij}^{\varphi}},$$

где k – номер муравья; P_{ij}^{k} – вероятность для k-го муравья поставить операцию j в позицию i расписания; f_{ij} – привлекательность варианта поставить операцию j в позицию i расписания на основании феромонного следа; e_{ij} – привлекательность варианта поставить операцию j в позицию i расписания согласно эвристике; α и β – коэффициенты значимости феромонного и эвристического факторов; N^k – множество операций, доступных для выполнения муравью k.

Если $f_{ij} = 0$, то присваиваем этой величине малое значение.

Выполнив все попытки в рамках одного поколения, мы должны пересчитать таблицу интенсивностей следов. Сначала все данные в ней умножаются на $E < 1$ (значение этого так называемого «коэффициента испарения» является одним из параметров алгоритма). Затем мы просматриваем все расписания, созданные «муравьями» последнего поколения. Рассмотрим расписание A со временем изготовления S_A. Если в этом расписании операция j занимала место i, то $f_{ij} = f_{ij} + Q / S_A$ (Q – произвольная константа, не влияющая на результат работы). Используем более эффективную разновидность алгоритма, в котором дополнительно усиливается след, оставленный лучшими «муравьями».

Модифицируем алгоритм, чтобы корректно учитывать наличие сборочных операций. Во время работы «муравья» мы храним: список L всех еще не выполненных операций; список W операций, выполнение которых может быть начато в текущий момент (из них мы делаем выбор на каждом шаге); список Q выполненных операций. Операции с порядковым номером 1 для каждой детали помещаются в W в начале работы. Выполнив операцию, мы переносим ее из W в Q, а следующую операцию над той же деталью добавляем в W. Если такой следующей операции не существует (деталь готова), выполняется проверка: если эта деталь – часть узла, все детали которого готовы, то только тогда первая операция над узлом добавляется в W. После этого узел рассматривается так же, как и деталь.

Материалы и методы

Разделение очередей как метод ускорения работы

Традиционный подход к решению задачи составления расписания методом муравьиной колонии основан на аналогии с задачей коммивояжера: выполняемые операции рассматриваются как аналог посещаемых вершин графа. Таким образом, на каждом шаге алгоритма при выборе очередной операции для помещения в очередь работ рассматривается все множество операций,

доступных для выполнения. Эта аналогия является достаточно точной в случае задачи с единственным станком, однако в случае цеха со многими станками это не так. Продемонстрируем это на простом примере.

Рассмотрим начальный момент времени. Если у нас в списке доступных операций W есть по одной операции $O_1 \ldots O_N$ для каждого из N станков $M_1 \ldots M_N$, то существует N! различных последовательностей, в которых мы можем поместить эти операции в расписание. При этом очевидно, что все полученные расписания будут идентичны: по одной операции на каждом из станков.

Таким образом, если считать, что каждая операция O_i может быть выполнена лишь на одном из станков M_j, то имеет смысл разбить множество W операций, доступных для выполнения, по типу станков, на которых они должны выполняться:

$W = \{W_1, W_2, \ldots W_N\}$, где N – число различных видов станков;

$W_i = \{O_{i1}, O_{i2}, \ldots O_{iz}\}$, где любая операция O_{ik} выполняется на станке M_i.

Благодаря этому мы сможем формировать порядок выполнения операций для каждого из этих множеств W_i по отдельности, применяя метод муравьиной колонии только к операциям из этого множества. Назовем получающиеся множества W_i входящими станочными списками задач.

Не будем забывать, что выполнение операции из одного входящего станочного списка задач может приводить к появлению следующей операции над той же деталью в другом входящем станочном списке. В связи с этим если мы просто будем просматривать каждый из входящих списков до исчерпания, а затем переходить к следующему, то следующие списки будут забиваться операциями, которые на нулевой момент времени не могли быть сделаны. Когда мы перейдем к следующему списку W_{i+1}, нам будет необходимо отсекать операции, которые не могли быть выполнены, например, в начальный момент времени, что снизит производительность алгоритма.

Чтобы избежать этого, введем в рассмотрение время.

Список операций, поставленных в расписание работ станка M_i, назовем станочной очередью Q_i. Длиной станочной очереди L_{Qi} станка M_i назовем момент планового окончания последней операции из станочной очереди Q_i.

Каждый раз, когда нам нужно выбрать операцию для помещения в расписание, мы сначала выбираем самую короткую станочную очередь Q_k: $L_{Qk} = \min(L_{Qi})$ и операцию из входящего списка W_k, относящемуся к тому же станку M_k.

Из этого вытекает еще одно изменение в алгоритме. У нас могут быть несколько станочных очередей равной длины (например, в начале работы, когда все станки свободны, все длины равны 0), поэтому сначала мы в любом порядке рассматриваем все кратчайшие входящие очереди, выбирая из них по одной операции, и только после этого добавляем во входящие списки все новые операции, полученные на этом шаге.

Классический метод:

Пока ДоступныеОперации не пусто
{
Сгенерировать вероятности выполнения каждой из операций;
Случайно выбрать операцию O_1 по сгенерированным вероятностям;
Убрать O_1 из ДоступныеОперации и НевыполненныеОперации;
Определить операцию O_2, ставшую доступной для выполнения;

Добавить O_2 в ДоступныеОперации;
}
Предлагаемый метод:
Пока ДоступныеОперации не пусто
{
Определить множество КратчайшиеОчереди;
Пока КратчайшиеОчереди не пусто
{
Выбрать любую из кратчайших очередей Q_i;
Сгенерировать вероятности выполнения каждой из операций соответствующего ВходящегоСписка W_i;
Случайно выбрать операцию O_1 по сгенерированным вероятностям;
Увеличить длину очереди Q_i на длительность операции O_1;
Убрать O_1 из ВходяшегоСписка W_i и НевыполненныхОпераций;
Определить операцию O_2, ставшую доступной для выполнения;
Добавить O_2 во ВременныйМассив;
}
Перенести операции из ВременныйМассив в соответствующие ВходящиеСписки W_k;
Очистить ВременныйМассив;
}

Соответственно, вместо одной таблицы следов муравьев мы будем хранить отдельные таблицы на каждый тип станков. В таблице следов муравьев храним порядковый номер, под которым указанная операция должна стоять в очереди заданий соответствующего станка. При этом следует учесть, что порядковый номер, под которым операция была взята из входящего списка, может отличаться от порядкового номера в расписании станка: если в расписании загрузки станка нашлось более раннее подходящее свободное время, в которое поместили рассматриваемую операцию, то стоит сразу зафиксировать это решение. Соответственно, мы формируем таблицу следов, рассматривая операции в порядке начала выполнения в очереди станка. Так как по условию задачи у нас может быть несколько идентичных станков, соответствующих одной входящей очереди, то при наличии нескольких операций с одинаковым временем начала (например, $T = 0$) всем им присваивается одинаковый номер в таблице следов. Использование в качестве значения следа суммы всех значений, включая более ранние (данный метод описан в работе [18]), устраняет возникающую проблему неоднозначности на стадии использования следов.

В результате сложность метода снижается с $O(N^2)$ до $O(n_1^2 + n_2^2 + \ldots n_k^2)$, $N = n_1 + n_2 + \ldots n_k$, что в случае крупных изделий и большого числа видов станков оказывается существенным. Помимо этого ускоряется обучение за счет улучшения соотношения сигнал/шум – у нас больше нет функционально идентичных решений, оставляющих разные следы.

Результаты

Для проверки предлагаемого метода были рассмотрены пять реальных производственных задач (структуры сборочных единиц, последовательности операций и их длительности и станочный парк). В отличие от многих задач

классических тестовых наборов, в этих изделиях распределение операций по типам очень неравномерное: обычно операции 3-4 самых популярных видов составляют примерно по 10 % от общего числа и занимают примерно такой же процент времени.

Было сделано по пять попыток для каждой из задач (табл. 1).

Таблица 1

Параметры рассматриваемых задач

Задача	Детали	Узлы	Уровни узлов	Операции
1	29	5	3	57
2	92	31	4	344
3	146	39	4	514
4	341	106	4	1133
5	607	138	5	2402

Параметры алгоритма:

Число поколений: 6. «Муравьев» в поколении: 100. α =1, β = 3, E = 0,9. Коэффициент элитного муравья: 3,33. Так как предметом рассмотрения является в первую очередь время работы алгоритма, то для нас несущественно, являются ли эти параметры оптимальными для рассматриваемых задач.

Эксперименты проводились без эвристики (вариант «Нет») и с «жадной» эвристикой ($e_j = C / T_j$, где T_j – длительность операции j; C – произвольная константа). Лучшее значение в рамках одной эвристики выделено в табл. 2 жирным шрифтом.

Таблица 2

Результаты

Задача	Эвристика: нет				Эвристика: жадная			
	Классический		Предлагаемый		Классический		Предлагаемый	
	Результат, ч	Время работы, с	Результат, ч	Время работы, с	Результат, ч	Время работы, с	Результат, ч	Время работы, с
1	29,916	84,6	**28,54**	**49,8**	**29,732**	123,8	32,418	**62**
2	64,228	3520	**62,082**	**343,2**	64,266	4271,6	**60,982**	**816,6**
3	178,296	4618,8	**175,518**	**866,6**	**167,89**	5102,8	170,218	**617,6**
4	763,85	42 736	**691,486**	**3436,2**	743,924	34 392,8	**729,202**	**2070,6**
	1532,112	88 226,8	**1511,634**	**4302**	1607,978	67 631,8	**1525,72**	**6126**

Обсуждение

Как можно заметить, во всех случаях предлагаемый алгоритм работал быстрее и почти всегда давал несколько лучшие результаты. Исключениями

по полученному результату являются задачи 1 и 3 при использовании «жадной» эвристики. В первом случае малые размеры подмножеств в сочетании с «жадной» эвристикой приводили к эффективному застреванию в локальных минимумах, не давая найти глобальный минимум. Из-за этого в задаче 1 сочетание «жадной» эвристики и предлагаемого метода стабильно выдавало худшие решения из четырех возможных сочетаний. Определение характеристик задач (общей размерности и распределения операций по станкам), которые делают случайный выбор предпочтительным по сравнению с использованием эвристики, может быть предметом дальнейших исследований.

В задаче 3 с эвристикой полученное относительно небольшое различие, по всей видимости, является следствием случайности: все попытки, предпринимаемые посредством обоих методов, давали примерно одинаковые результаты в диапазоне 167–172, но в одной из попыток классическим методом был получен результат 161,28.

Для крупных задач число муравьев было явно недостаточным, потому полученные результаты имеют довольно случайный характер. Свидетельством этого является тот факт, что в некоторых случаях решение, в итоге ставшее лучшим, было найдено в первом-втором циклах попыток, и дальнейшее самообучение улучшало средний результат по набору попыток, но не лучший. Очевидно, фактически числа попыток было недостаточно для обучения модели. Отметим, что при этом использование эвристики позволяло улучшить результаты.

Так или иначе, поставленная цель достигнута: на крупных задачах экономия времени весьма существенна, и сэкономленный ресурс может быть использован для увеличения числа попыток.

Дальнейшее увеличение эффективности работы может быть достигнуто за счет использования многопоточности. В данном случае она не применялась, во-первых, для обеспечения корректного сравнения именно алгоритмической эффективности методов, а во-вторых – из-за ограничений используемой вычислительной среды. В текущей реализации алгоритма одинаковые кратчайшие очереди рассматриваются последовательно, но так как поиски решения для них независимы, то их можно осуществлять в параллельных потоках. Кроме того, если какая-то очередь длиннее кратчайшей очереди на величину, меньшую типичной длины операции, то, вероятно, ее тоже можно обрабатывать в параллельном потоке.

Кроме того, улучшить получаемые результаты именно для задачи с многоуровневыми сборками можно, используя специальную структурную эвристику. Пример такой эвристики был нами рассмотрен в работе [19].

Заключение

В настоящем исследовании предложена модификация метода муравьиной колонии применительно к задаче составления расписания производства многостаночного цеха единичного производства изделий, включающих многоуровневые сборки. Показано, что разделение входящей очереди операций на очереди меньшего размера в соответствии с типами станков, на которых должны выполняться указанные операции, существенно сокращает время работы алгоритма. При этом получаемые результаты на крупных задачах, как правило, не хуже, чем достигаемые классическим методом.

Список литературы

1. Fry T. D., Olife M. D., Minor E. D., Keong Leong G. The effect of product structure and sequencing rule on assembly shop performance // International Journal of Production Research. 1989. № 27 (4). P. 671–686.
2. Philipoom P. R., Russel R. S. L., Fry T. D. A preliminary investigation of multi-attribute based sequencing rules for assembly shops // International Journal of Production Research. 1991. № 29 (4). P. 739–753.
3. Doctor S. R., Cavalier T. M., Egbelu P. J. Scheduling for machining and assembly in a job-shop environment // International Journal of Production Research. 1993. № 31 (6). P. 1275–1297.
4. McCoy D. H. C., Egbelu P. J. Minimizing production flow time in a process and assembly job shop // International Journal of Production Research. 1998. № 36 (8). P. 2315–2332.
5. Park M. W., Kim Y. D. A heuristic algorithm for a production planning problem in an assembly system // Journal of Operational Research Society. 1999. № 50. P. 138–147.
6. Moon-Won P., Kim Y. D. A branch and bound algorithm for a production scheduling problem in an assembly system under due date constraints // European Journal of Operational Research. 2000. № 123 (3). P. 504–518 .
7. Reeja M. K., Rajendran Ch. Dispatching rules for scheduling in assembly jobshops // International Journal of Production Research. 2000. № 38 (9). P. 2051–2066.
8. Cao W., Xie Z., Yang J. [et al.]. A Reverse Order Hierarchical Integrated Scheduling Algorithm Considering Dynamic Time Urgency Degree of the Process Sequences // Electronics. 2022. № 11. doi: 10.3390/electronics11121868
9. Omkumar M., Shahabudeen P. Ant Colony Optimisation for multi-level assembly job shop scheduling // International Journal of Manufacturing Research. 2009. № 1. P. 127–152.
10. Dorigo M., Maniezzo V., Colorni A. Ant system: Optimization by a colony of cooperating agents // IEEE Trans. Systems Man Cybernet. 1996. Part B26 (1). P. 29–41.
11. Sasikala B., Eswaramurthy V. P. Ant Colony Optimization for Job Shop Scheduling Problem Using Priority Rules // International Journal of Engineering and Techniques. 2017. Vol. 3, is. 6.
12. Abidia M. H., Al-Harkanb I., El-Tamimib A. M. [et al.]. Ant Colony Optimization for Job Shop Scheduling to Minimize the Total Weighted Tardiness // Proceedings of the 2014 Industrial and Systems Engineering Research Conference. 2013.
13. Huang M., Guo D., Guo F. An Improved Ant Colony Algorithm for Multi-objective Flexible Job-shop Scheduling Problem // 2022 IEEE 10th International Conference on Computer Science and Network Technology (ICCSNT). 2022. P. 1–3. doi: 10.1109/ICCSNT56096.2022.9972994
14. Chen F., Xie W., Ma J. Textile flexible job-shop scheduling using PACO algorithm // Applied Sciences. 2024. Vol. 14 (9). doi: 10.3390/app14104082
15. Rahman M. L., Sazol A. Application of an improved ant colony optimization algorithm of hybrid strategies using scheduling for patient management in hospitals // Heliyon. 2024. Vol. 10 (21). doi: 10.1016/j.heliyon.2024.e40134
16. Matrenin P. V. Improvement of Ant Colony Algorithm Performance for the Job-Shop Scheduling Problem Using Evolutionary Adaptation and Software Realization Heuristics // Algorithms. 2023. № 16. P. 15. doi: 10.3390/a16010015
17. Tang Q., Wang H. Data-driven automated job shop scheduling optimization considering AGV obstacle avoidance // Sci Rep. 2025. № 15. doi: 10.1038/s41598-024-82870-1
18. Merkle D., Middendorf M. An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness Problems // Real-World Applications of Evolutionary Computing. EvoWorkshops 2000. Lecture Notes in Computer Science. № 1803. Springer, Berlin, Heidelberg. doi: 10.1007/3-540-45561-2_28

19. Иванов М. А. Структурная эвристика для построения расписания цеха разового производства со сборочными операциями методом муравьиной колонии // Проблемы управления в социально-экономических и технических системах : материалы XX Междунар. науч.-практ. конф. Саратов : Наука, 2024. 702 с

References

1. Fry T.D., Olife M.D., Minor E.D., Keong Leong G. The effect of product structure and sequencing rule on assembly shop performance. *International Journal of Production Research*. 1989;(27):671–686.
2. Philipoom P.R., Russel R.S.L., Fry T.D. A preliminary investigation of multi-attribute based sequencing rules for assembly shops. *International Journal of Production Research*. 1991;(29):739–753.
3. Doctor S.R., Cavalier T.M., Egbelu P.J. Scheduling for machining and assembly in a job-shop environment. *International Journal of Production Research*. 1993;(31):1275–1297.
4. McCoy D. H. C., Egbelu P.J. Minimizing production flow time in a process and assembly job shop. *International Journal of Production Research*. 1998;(36):2315–2332.
5. Park M.W., Kim Y.D. A heuristic algorithm for a production planning problem in an assembly system. *Journal of Operational Research Society*. 1999;(50):138–147.
6. Moon-Won P., Kim Y.D. A branch and bound algorithm for a production scheduling problem in an assembly system under due date constraints. *European Journal of Operational Research*. 2000;(123):504–518 .
7. Reeja M.K., Rajendran Ch. Dispatching rules for scheduling in assembly jobshops. *International Journal of Production Research*. 2000;(38):2051–2066.
8. Cao W., Xie Z., Yang J. et al. A Reverse Order Hierarchical Integrated Scheduling Algorithm Considering Dynamic Time Urgency Degree of the Process Sequences. *Electronics*. 2022;(11). doi: 10.3390/electronics11121868
9. Omkumar M., Shahabudeen P. Ant Colony Optimisation for multi-level assembly job shop scheduling. *International Journal of Manufacturing Research*. 2009;(1):127–152.
10. Dorigo M., Maniezzo V., Colorni A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Systems Man Cybernet*. 1996;(B26):29–41.
11. Sasikala B., Eswaramurthy V.P. Ant Colony Optimization for Job Shop Scheduling Problem Using Priority Rules. *International Journal of Engineering and Techniques*. 2017;3(6).
12. Abidia M.H., Al-Harkanb I., El-Tamimib A.M. et al. Ant Colony Optimization for Job Shop Scheduling to Minimize the Total Weighted Tardiness. *Proceedings of the 2014 Industrial and Systems Engineering Research Conference*. 2013.
13. Huang M., Guo D., Guo F. An Improved Ant Colony Algorithm for Multi-objective Flexible Job-shop Scheduling Problem. *2022 IEEE 10th International Conference on Computer Science and Network Technology (ICCSNT)*. 2022:1–3. doi: 10.1109/ICCSNT56096.2022.9972994
14. Chen F., Xie W., Ma J. Textile flexible job-shop scheduling using PACO algorithm. *Applied Sciences.* 2024;14. doi: 10.3390/app14104082
15. Rahman M.L., Sazol A. Application of an improved ant colony optimization algorithm of hybrid strategies using scheduling for patient management in hospitals. *Heliyon*. 2024;10(21). doi: 10.1016/j.heliyon.2024.e40134
16. Matrenin P.V. Improvement of Ant Colony Algorithm Performance for the Job-Shop Scheduling Problem Using Evolutionary Adaptation and Software Realization Heuristics. *Algorithms*. 2023;(16):15. doi: 10.3390/a16010015
17. Tang Q., Wang H. Data-driven automated job shop scheduling optimization considering AGV obstacle avoidance. *Sci Rep*. 2025;(15). doi: 10.1038/s41598-024-82870-1
18. Merkle D., Middendorf M. An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness Problems. *Real-World Applications of Evolutionary Computing.*

EvoWorkshops 2000. Lecture Notes in Computer Science. № 1803. Springer, Berlin, Heidelberg. doi: 10.1007/3-540-45561-2_28

19. Ivanov M.A. Structural heuristics for scheduling a one-time production workshop with assembly operations using the ant colony method. *Problemy upravleniya v sotsial'no-ekonomicheskikh i tekhnicheskikh sistemakh: materialy XX Mezhdunar. nauch.-prakt. konf = Management problems in socio-economic and technical systems: materials of the XX International Scientific and Practical Conference*. Saratov: Nauka, 2024:702. (In Russ)

Информация об авторах / Information about the authors

Михаил Андреевич Иванов
ведущий программист,
НефтемашСистема
(Россия, г. Саратов, ул. Гвардейская, 2А)
E-mail: mikeaivanov@gmail.com

Mikhail A. Ivanov
Lead programmer,
NeftemashSistema
(2A Gvardeiskaya street, Saratov, Russia)

Автор заявляет об отсутствии конфликта интересов /
The author declares no conflicts of interests.

Поступила в редакцию/Received 10.12.2024
Поступила после рецензирования/Revised 09.06.2025
Принята к публикации/Accepted 18.07.2025