Эволюционный алгоритм задачи составления расписания выполнения заказов клиентов с двумя критериями

Захаров А.О.
Омский филиал Института математики им. С.Л. Соболева СО РАН

13 мая 2025 г.

Аннотация

В данной работе рассматривается двухкритериальная задача составления расписания выполнения заказов клиентов. Строится эволюционный алгоритм на базе алгоритма SEMO. Проводится вычислительный эксперимент на серии тестовых примеров с тремя операторами мутации.

1 Введение

Рассматривается задача составления расписания выполнения заказов m клиентов. У каждого клиента есть только один заказ, который включает в себя набор продуктов. Необходимо выполнить заказы всех клиентов, т.е. произвести для них продукты. Зададим множество всех клиентов через $M = \{1, \ldots, m\}$, множество всех продуктов через $N = \{1, \ldots, n\}$. В общем случае заказ каждого клиента состоит из некоторого подмножества множества продуктов $N_i \subset N \ \forall i \in M$. Мы будем рассматривать $N_i = N$.

Продукт j характеризуется длительностью производства $p_{ij} \geq 0$ для клиента i и величиной начальной переналадки $s_j \geq 0$. При переходе от производства продукта j к продукту j' требуется переналадка, которая имеет длительность $s_{jj'} \geq 0$. Здесь $\forall i \in M$, $\forall j, j' \in N$, $j \neq j'$.

Как правило, в качестве критериев задачи составления расписания берутся функции ΣC_j (минимизация суммы моментов завершения выполнения заказов клиентов), C_{max} (минимизация момента завершения выполнения заказа последнего клиента), максимизация суммы весов заказов, выполненных в срок. Например, первый и третий из них рассматривались в работе [1]. Была поставлена двухкритериальная задача с критериями ΣC_j и C_{max}, получена верхняя оценка на мощность множества Парето и описан конструктивный алгоритм его построения в [2]. Однако данный алгоритм является трудоемким, т.к. заключается в поиске оптимального решения серии однокритериальных задач с критерием ΣC_j. Поэтому в данной статье

предлагается рассмотреть эволюционный алгоритм на базе схемы алгоритма SEMO [3].

2 Постановка задачи и алгоритм

Расписание будем задавать в виде перестановки операций $\pi = (\pi_1, \ldots, \pi_{nm})$, $\pi_k = (i, j)$, где пара (i, j) означает производство продукта j для клиента i. Множество всех возможных перестановок обозначим через Π. Векторный критерий $f = (f_1, f_2)$ рассматривается на "минимум"

$$f_1(\pi) = \Sigma_{i \in M} C_i(\pi), \quad f_2(\pi) = \max_{i \in M} C_i(\pi),$$

где $C_j(\pi)$ – момент завершения заказа клиента i на перестановке π, $i \in M$. Напомним, что множество Парето $P_f(\Pi)$ – это множество недоминируемых перестановок в смысле отношения $\leq$ по критерию f (в случае задачи на минимум), т. е. $P_f(\Pi) = \{\pi \in \Pi \mid \nexists \pi' \in \Pi : \ f(\pi') \leq f(\pi)\}$. Причем, два вектора $y' = (y'_1, y'_2), y'' = (y''_1, y''_2)$ находятся в отношении $y' \leq y''$, если $y'_i \leqslant y''_i$, $i \in \{1, 2\}$, $y' \neq y''$.

Как было сказано выше, будем искать аппроксимацию множества Парето используя эволюционный алгоритм на базе алгоритма SEMO. Особью является перестановка операций π. Шаги алгоритма представлены ниже.

1. Сгенерировать особь π случайным образом. Положить $\tilde{P}_f(\Pi) = \{\pi\}$.
2. Повторить шаги 3–5 до выполнения критерия остановки.
3. Случайным образом выбрать особь π' из множества $\tilde{P}_f(\Pi)$.
4. Применить оператор мутации к особи π', получить потомка $\pi'' = Mut(\pi')$.
5. Если во множестве $\tilde{P}_f(\Pi)$ не существует перестановки, которая доминирует перестановку π'', то положить $\tilde{P}_f(\Pi) = \tilde{P}_f(\Pi) \cup \{\pi''\}$ и удалить из множества $\tilde{P}_f(\Pi)$ все такие перестановки, которые доминируются перестановкой π''.
6. Результатом работы алгоритма будет аппроксимация множества Парето $\tilde{P}_f(\Pi)$.

На шагах 1, 3 особь генерируется и выбирается псевдослучайным образом с равномерным распределением. На шаге 4 рассмотрены следующие варианты операторов мутации: сдвиг, обмен и с тяжелыми "хвостами" со степенным распределением. В качестве критерия остановки использовалось ограничение на число итераций.

3 Вычислительный эксперимент

Реализация алгоритма была написана на языке C++ и был проведен вычислительный эксперимент на сериях тестовых задач, которые использовались в экспериментах для однокритериальной задачи из работ [1, 4]. Они строились по правилу, приведенному в [5], значения количества клиентов m и количества продуктов n брались из множества $\{5, 10, 15, 20\}$, переналадки генерировались равномерно из интервалов [10, 20], [1, 30], [25, 35], [15, 45], длительности – из интервала [1, 15].

На каждой тестовой задаче выполнялся запуск трех конфигураций (три оператора мутации) по 30 раз каждая конфигурация. Запуск останавливался, если достигалось ограничение в 1 млн. итераций.

Результаты эксперимента приведены в таблице 1. По строкам приводятся агрегированные результаты по 30 запускам по серии с указанными значениями количества клиентов m и количества продуктов n. В каждой серии 4 задачи. По столбцам приводятся результаты в разрезе трех кофигураций (трех мутаций) - мутация с тяжелыми "хвостами" со степенным распределением (power), сдвиг (shift) и обмен (swap). Показатель cardinality – среднее число точек в аппроксимации множества Парето. Характеристику rate мы рассчитывали следующим образом. По каждой задаче из серии брали результаты 30 запусков с тремя конфигурациями, выбирали парето-оптимальные точки. В качестве rate брали отношение количества задач в серии, где конфигурация дает хотя бы одну парето-оптимальную точку, к количеству задач в серии. Данный показатель можно рассматривать, как вклад конфигурации в формирование аппроксимации множества Парето.

m	n	cardinality			rate		
		power	shift	swap	power	shift	swap
10	5	1,1	1,3	1,125	0,75	0,5	0,5
10	10	1	1,46	1,13	0,25	0,5	0,5
10	15	1	1,4	1,13	0	0,5	0,75
10	20	1	1,475	1,11	0,25	0,5	0,5
20	5	1	1,43	1,085	0,5	0,75	0,5
20	10	1,15	1,4	1,145	0,75	0,5	0,5
20	15	1,2	1,5	1,12	0	0,75	0,5
20	20	1	1,5	1,1	0,5	0,5	0,5
20	50	1,125	1,7	1,015	0,5	0,5	0,25

Таблица 1. Средние значения мощности аппроксимации множества Парето (cardinality) и степени попадания конфигурации в аппроксимацию множества Парето (rate)

В целом результаты эксперимента показывают, что использование мутации сдвига дает большее количество точек в аппроксимации множества Парето, по сравнению с двумя другими мутациями. Мутация с тяжелыми

”хвостами“ часто давала одну точку на одном запуске. В части вклада мутации в формирование аппроксимации множества Парето не наблюдается доминирование какой-то одной конфигурации.

В дальнейшем планируется добавить операторы кроссинговера в эволюционный алгоритм. Также провести эксперимент на других сериях тестовых примеров, провести сравнение полученных результатов с результатами альтернативных алгоритмов.

Благодарности

Работа выполнена при финансовой поддержке РНФ, грант N 22-71-10015.

Список литературы

[1] Zakharova, Y.V., Zakharov, A.O. (2024). Integer Programming Models and Metaheuristics for Customer Order Scheduling. In: Eremeev, A., Khachay, M., Kochetov, Y., Mazalov, V., Pardalos, P. (eds) Mathematical Optimization Theory and Operations Research: Recent Trends. MOTOR 2024. Communications in Computer and Information Science, vol 2239. Springer, Cham.

[2] Захаров А.О. Анализ решений задачи составления расписания выполнения заказов клиентов с двумя критериями. Труды XX международной научной конференции "Проблемы теоретической кибернетики". 5–8 декабря 2024 г, Москва (в печати).

[3] Laumanns M., Thiele L., Zitzler E., Welzl E., Deb K. Running time analysis of a multi-objective evolutionary algorithm on a simple discrete optimization problem // Parallel Probl. Solving from Nature. 2002. V. 2439. Berlin: Springer, 2002. P. 44–53.

[4] Borisovsky P.A. , Zakharov A.O. , Zakharova Y.V. Evolutionary Algorithms for Customer Order Scheduling. Известия Иркутского государственного университета. Серия: Математика (Bulletin of Irkutsk State University. Series Mathematics). 2025 (в печати).

[5] Hazır, O., Gunalay, Y., Erel, E.: Customer order scheduling problem: a compara- tive metaheuristics study. The International Journal of Advanced Manufacturing Technology 37, 589–598 (2008).