УДК 519.8

DOI: 10.21779/2542-0321-2020-35-4-34–39

***В.С. Георги***

## Составление расписаний многопроцессорных машин

*Дагестанский государственный университет; Россия, 367000, г. Махачкала, ул. М. Гаджиева, 43а; georgywaleed1993@hotmail.com*

В статье рассматриваются составление расписаний на многопроцессорных машинах с использованием алгоритма наибольшего времени обработки, последовательность действий в данном алгоритме и причины, по которым данный алгоритм является актуальным и широко используемым. Также рассматриваются составление оптимальных расписаний, которые одновременно называются расписаниями с наименьшим временем обработки, и их связь с расписаниями, составленными первым алгоритмом. В обоих случаях показаны примеры составления простых расписаний. Представлены математические уравнения для удостоверения в правильности составленных расписаний, а также в возможности составления более комплексных расписаний. В статье объяснены основные термины и определена основная проблема данной задачи. Представляются результаты исследований, которые ранее проводились учеными в данной сфере. Все вышеперечисленное было проведено с использованием диаграмм и математических уравнений.

Ключевые слова: *LPT расписание, OPT расписание, задача, машина, время выполнения.*

### Введение

Составление расписаний многопроцессорных машин – это трудная оптимизационная задача. Расписание – указание, в какое время и на каких машинах должны выполняться задачи.

Многопроцессорные планировщики должны составить расписание задач, которые могут зависеть или не зависеть друг от друга. Например, чтение учетных данных пользователя из консоли, затем использование этих данных для аутентификации и в случае, если аутентификация прошла успешно, отображение некоего сообщения на консоли. В данной ситуации очевидно, что каждая задача зависит от предыдущей и что между задачами существует некий порядок выполнения. Такую последовательность задач можно смоделировать при помощи частичного упорядочения. Тогда по определению множество задач составляют решетчатую структуру.

Многопроцессорные алгоритмы составления расписаний бывают статическими или динамическими. Алгоритм является статическим, если решение выполнения таких-то задач на таких-то процессорах будет приниматься до запуска программы. Алгоритм является динамическим, если данное решение будет принято во время выполнения программы. Для алгоритмов статического составления расписаний типичным подходом является классифицирование задач в соответствии с их приоритетами и использованием техники планирования списков для распределения их на процессоры.

### Составление расписаний многопроцессорных машин

Простым, часто используемым алгоритмом составления расписаний, является алгоритм *LPT* (*Longest Processing Time*), который сортирует задачи по времени их обработки (сначала обрабатывается самая длинная задача), а затем назначает машине задачи

с самым ранним временем окончания обработки. Алгоритм *LPT* определяет приоритеты задач, которые должны выполняться в соответствии с порядком времени их обработки.

Одной из фундаментальных проблем является составление расписаний выполнения *n* независимых задач на $m \geq 2$ параллельных процессорах, чтобы минимизировать общее время выполнения расписания. Для этой задачи Грэм предложил очень простой аппроксимационный эвристический алгоритм, известный как планирование списков *LS* (*List Scheduling*), который сначала помещает задачу в список приоритетов, а затем назначает первую задачу из этого списка первому доступному процессору. Он доказал, что в худшем случае такая эвристика имеет производительность $(2 - \frac{1}{m})$.

Наблюдения за планированием списков показали, что худший случай возникает тогда, когда задача, которая завершает расписание, является длинной. Грэм улучшил планирование списка, выбрав конкретный начальный список приоритетов – последовательность *LPT*, которая завершается кратчайшими заданиями. Он показывает, что наихудший коэффициент производительности эвристики *LPT* равен $(\frac{4}{3} - \frac{1}{3m})$. Тем не менее, эксперименты показывают, что эвристика *LPT* на практике работает намного лучше, чем коэффициент производительности в худшем случае, особенно при увеличении числа задач. Следовательно, Коффман и Сетхи обобщили алгоритм *LPT* Грэма, включив в него параметр, характеризующий количество задач, назначенных на процессор *LPT*. Они показывают, что если расписание *LPT* имеет конечную задачу, которая выполняется на процессоре с, как минимум *k–1*, другими задачами (при условии $k \geq 3$), то наихудший коэффициент производительности *LPT* фактически равен $\frac{(k+1)}{k} - \frac{1}{(km)}$.

С точки зрения производительности Фрэнк и Ринной Кан доказали, что *LPT* асимптотически абсолютно и относительно оптимален, даже если процессоры имеют разные скорости. При такой хорошей производительности эвристика *LPT* была успешно применена к различным средам планирования, составляя расписание с минимальным временем выполнения. Например, в многоэтапном планировании с параллельными машинами, где каждое задание должно обрабатываться в несколько этапов в разные периоды времени, а каждый этап состоит из нескольких параллельных компьютеров, *LPT* применяется на каждом отдельном этапе. При параллельном планировании, когда задания поступают с течением времени, а время существования и обработки каждой задачи неизвестно до ее поступления, планирование *LPT* используется между любыми двумя последовательными моментами поступления.

Что еще более важно, благодаря превосходному балансу между производительностью и эффективностью эвристики, *LPT* стал главным критерием для разработки алгоритмов эвристического планирования высокого качества.

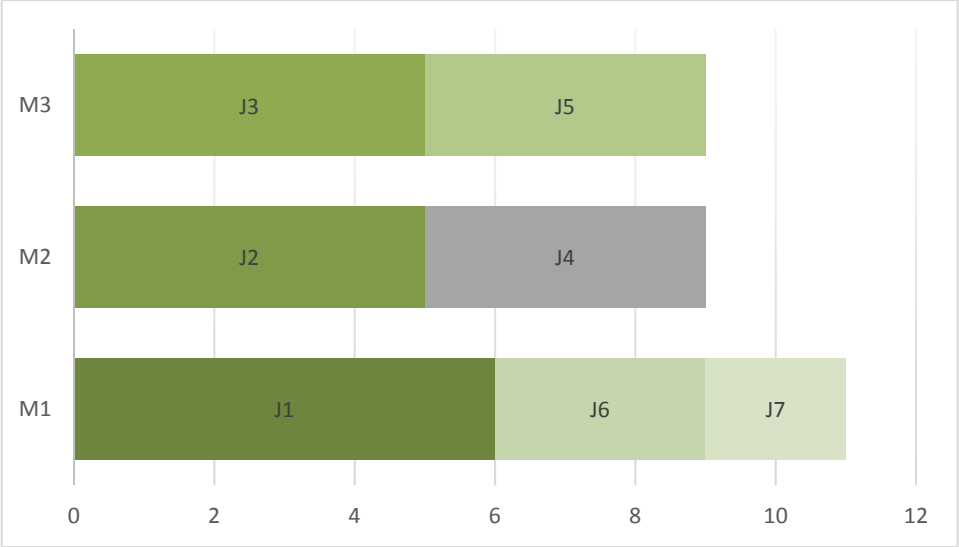
Предположим, у нас есть *n* задачи, каждая из которых имеет время обработки $p_1$, $p_2$, ..., $p_n$. Также имеется *m* одинаковых машин (процессоров). Наша цель – составить расписание обработки данных задач на машинах (процессорах) так, чтобы минимизировать рабочий диапазон (длину) составленного расписания.

Первым шагом в *LPT* является упорядочение заданного списка задач в порядке убывания времени их выполнения. Затем составляется расписание выполнения этих задач в данном порядке. Каждая новая задача будет назначена процессору, время выполнения задачи которого является минимальным.

Допустим, у нас есть 3 машины и 7 задач со временем обработки {2, 4, 6, 3, 4, 5, 5}. После упорядочения в порядке убывания времени обработки – {6, 5, 5, 4, 4, 3, 2}. На

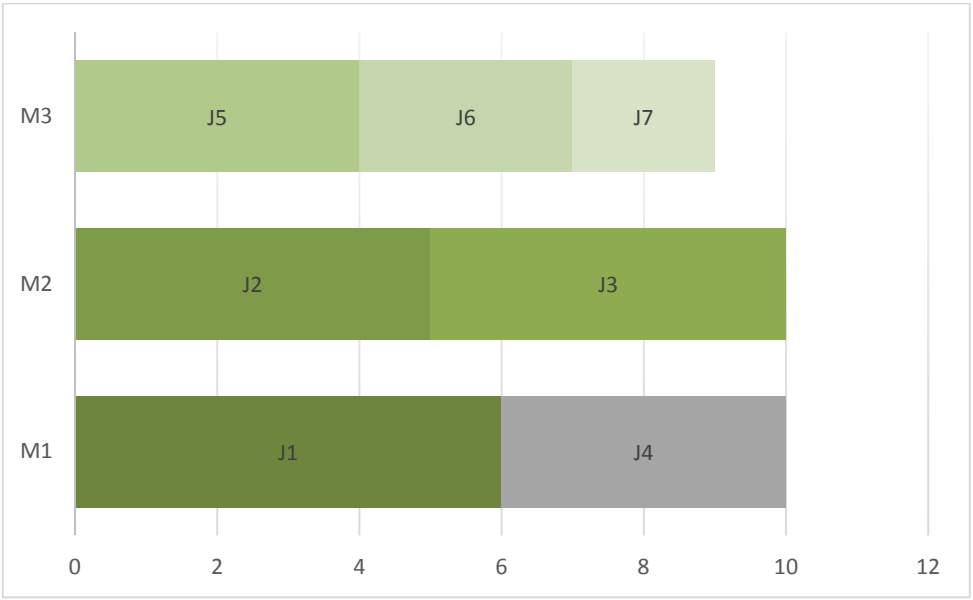
момент начала обработки задач все машины являются незанятыми, поэтому возможно назначить первую задачу со временем обработки 6 на любую машину.

Следующая задача также имеет время обработки 6 и назначается на вторую машину (можно также на третью). Затем назначается третья задача со временем обработки 5 на третью машину (или на вторую в случае, если предыдущая задача была назначена на третью машину). Выполнение алгоритма продолжается, пока не разместятся все задачи на всех машинах. При завершении получаем следующее расписание:




Как видно, *LPT* распределяет каждую задачу так, что время обработки задач по данному расписанию составляет 11 единиц времени (6 + 3 + 2).

Интересно отметить, что данное решение не является оптимальным. Оптимальное решение будет иметь наименьшее возможное время выполнения всех задач. Поэтому данное расписание можно составить следующим образом:




По данному расписанию общее время обработки задач составляет 10 единиц времени (сохранена 1 единица времени).

Время выполнения по оптимальному расписанию имеет нижнюю границу $OPT \geq \sum p_i/m$. В данном случае $\sum p_i/m$ = 28/3 = 9.33, а поскольку $OPT$ = 10, то данное правило соблюдено.

Пусть $TS$ будет набором, состоящим из $n$ независимых задач, а $m$ будет количеством машин. Доказано, что $t_{LPT}(TS, m) / t_{opt}(TS, m) \leq (\frac{4}{3} - \frac{1}{3m})$. Следовательно,

$$t_{LPT}(TS, m) \leq (\frac{4}{3} - \frac{1}{3m})\, t_{opt}(TS, m), \quad (1)$$

где $t_{LPT}(TS, m)$ и $t_{opt}(TS, m)$ обозначают $C_{max}$ $LPT$ и оптимального графика соответственно. Данное уравнение верно для любого количества машин $m$.

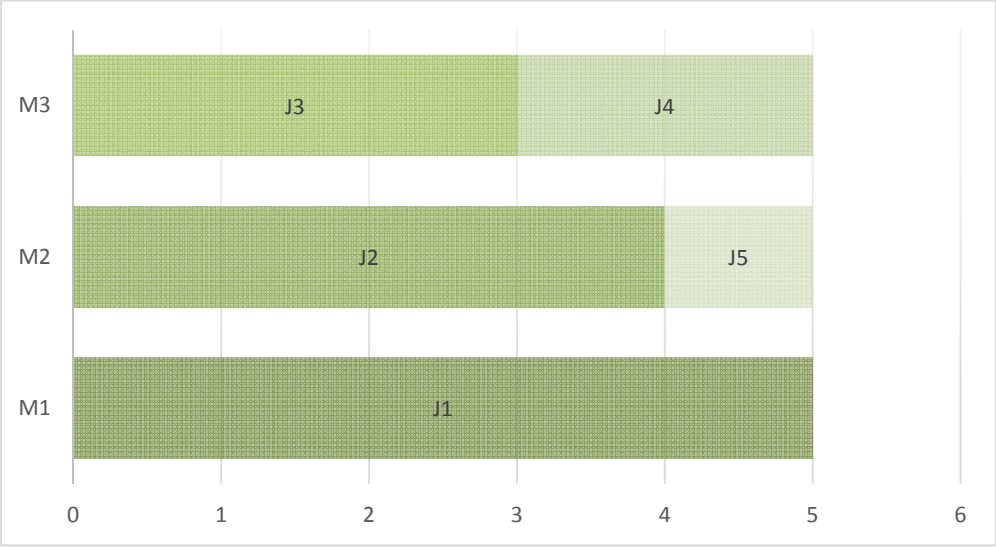
Приведем пример: $m$ = 3, $J$ = 5, 5, 4, 4, 3, 3, 3. В данном случае $t_{LPT}(TS, m) = 11$, $t_{opt}(TS, m) = 9$, $m$ = 3. Из (1) следует $11 \leq (\frac{4}{3} - \frac{1}{9})9$, т. е. $11 \leq 11$, что верно.

Основываясь на предыдущем примере, когда имеется нечетное количество задач $2m + 1$ (при условии, что $m \geq 1$) и

$J_1 = J_2 = 2m\text{-}1,\ J_3 = J_4 = 2m\text{–}2,\ J_5 = J_6 = 2m\text{–}3 \ldots J_{m-1} = J_m = J_{m+1} = m,$

можно заметить, что в $LPT$ выполнение последней задачи $J_{2m+1}$ начинается в $3m$–1 и заканчивается в $4m$–1, а в случае оптимального расписания $OPT$ все задачи окончательно выполняются в $3m$.

Если в $LPT$ расписании каждая машина обрабатывает не более двух задач, то оно является оптимальным, то есть $t_{LPT}(TS, m) = t_{opt}(TS, m)$. Например, $m$ = 3, $p$ = 5, 4, 3, 2, 1.




Данное расписание по $LPT$ также является оптимальным.

Было доказано: если время выполнения $p_l$ (также являющееся наименьшим временем выполнения $p_{min}$) последней задачи $l$ больше, чем $OPT$/3 ($p_l > \frac{OPT}{3}$), то время выполнения расписания по $LPT$ равно $OPT$. Иначе: если $p_l \leq \frac{OPT}{3}$, то можно использовать следующее уравнение:

$$C^S_{max} \leq OPT + p_l\left(1 - \frac{1}{m}\right) \leq \left(\frac{4}{3} - \frac{1}{3m}\right) OPT. \quad (2)$$

Например, имеются $m$ = 2 процессора и список задач со временем обработки {4, 4, 3, 3, 3}. $LPT$ составит расписание, где $C^S_{max} = 10$, $OPT$ = 9, задачи 1 и 2 будут обраба-

тываться на машине 1, а остальные задачи – на машине 2. Опираясь на (2), получаем $10 \leq 10.5 \leq 10.5$, что подтверждает правильность данного уравнения.

На основании проведенного исследования была доказана возможность использования уравнений, которые помогают удостовериться в правильности составленного расписания и времени его выполнения при помощи простых примеров. В дальнейшем можно модифицировать данные уравнения для применения к более сложным расписаниям, написать программу для их автоматического составления и использовать эти данные для решения других задач, например составления школьных расписаний.

**Литература**

1. *Garey M.R., Johnson D.S.* Computers and Intractability: A Guide to the Theory of NP-Completeness. – N. Y.: W.H. Freeman and Company, 1979. – P. 238.

2. *Yu-Kwong K., Ishfaq A.* Static scheduling algorithms for allocating directed task graphs to multiprocessors // ACM Computing Surveys. – 1999. – Vol. 31, № 4. – P. 471.

3. *Coffman Jr.E.G., Sethi R.* A generalized bound on LPT sequencing // Revue Française d'Automatique Informatique: Recherche Operationnelle. – 1976. – № 10. – P. 310.

4. *Garey M.R., Johnson D.S.* Strong NP-completeness results: Motivation, examples and implications // J. Assoc. Comput. – New Jersey, 1978. – № 25. – P. 508.

5. *Baruah S., Bertogna M., Buttazzo G.* Multiprocessor Scheduling for Real-time Systems // Springer International Publishing. Switzerland. – 2015. – № 5. – P. 234.

6. *Gaitan M.G., Yomsi P.M.* Multiprocessor Scheduling meets the Industrial Wireless // Research Gate. – 2019. – Vol. 5, № 1. – P. 59–76.

7. *Al-Salami K.H., Sawadi Z.* Task Scheduling for Multiprocessor Systems Using Queueing Theory // ResearchGate. – 2016. – Vol. 7, № 2. – P. 24–33.

8. *Vijai Sai R., Lavanya M., Srinivasan B.* // International Journal of Pure and Applied Mathematics. – 2018. – Vol. 118, № 20. – P. 3149–3155.

9. *Younes A., Ben Salah A., Farag T., Alghamdi F.A., Badawi U.A.* // Journal of Theoretical and Applied Information Technology. – 2019. – Vol. 97, № 12. – P. 3477–3487.

10. *Orr James & Baruah Sanjoy.* Multiprocessor scheduling of elastic tasks // RTNS '19: Proceedings of the 27th International Conference on Real-Time Networks and Systems. – 2019. – P. 133–142.

*Поступила в редакцию 29 июня 2020 г.*

UDC 519.8

DOI: 10.21779/2542-0321-2020-35-4-34–39

**Multiprocessors Scheduling**

***V.S. Giorgi***

*Dagestan State University; Russia, 367000, Makhachkala, M. Gadzhiev st., 43a; georgywaleed1993@hotmail.com*

This article discusses such issues as scheduling on multiprocessor machines using the longest processing time algorithm; the sequence of actions in this algorithm and the reasons why it is relevant

and widely used based on its efficiency and productivity; the compilation of optimal schedules, which are simultaneously called schedules with the least processing time, and their relationship with schedules compiled by the first algorithm. In both cases, examples of simple schedules were shown. Mathematical equations were presented to verify the correctness of the schedules, as well as the possibility of compiling more complex schedules. Basic terms were explained, and the main problem was determined. The results of studies that were previously conducted by scientists in this field, and how their results helped in solving similar problems are presented. All of the above was carried out using diagrams and mathematical equations.

Keywords: *LPT scheduling, OPT scheduling, task, machine, completion time.*

*Received 29 June's 2020*