УДК 519.854.2

СОСТАВЛЕНИЕ РАСПИСАНИЙ С МИНИМИЗАЦИЕЙ СУММАРНОГО ЗАПАЗДЫВАНИЯ НА ОДНОМ ПРИБОРЕ МЕТОДОМ ПАРАЛЛЕЛЬНЫХ МУРАВЬИНЫХ КОЛОНИЙ

Долгова О. Э. – асп., стажер-исследователь, e-mail: edyardovna@gmail.com; *Пересветов В. В.* – канд. ф.-м. наук, с.н.с., e-mail: vvperesv@yandex.ru (ВЦ ДВО РАН)

Разработан метаэвристический муравьиный алгоритм с несколькими параллельными независимыми колониями для решения задачи составления расписаний обслуживания множества требований на одном приборе с критерием минимизации суммарного взвешенного запаздывания. Предложена эффективная комбинированная схема локального поиска с последовательным применением стохастических методов. Показаны преимущества параллельного поиска решения муравьиными колониями, которые заключаются в более высокой скорости сходимости и стабильности времени получения оптимальных расписаний. Приводятся результаты расчетов с использованием технологии параллельных вычислений MPI.

The ant colony optimization metaheuristic algorithm with several parallel independent ant colonies has been worked out for solving the scheduling problem of processing a set of jobs on a single machine with the criterion of minimizing the total weighted delay. The effective complex local-search scheme with sequential application of stochastic methods has been proposed. The advantages of the parallel ant colonies' search for the optimal solution are shown to consist in a higher rate of the convergence and stability of the time required for deriving optimal schedules. The results of calculations with the use of the parallel computing MPI technology are given.

Ключевые слова: составление расписаний для одного прибора, минимизация суммарного взвешенного запаздывания, параллельные муравьиные колонии.

Рассматривается NP-трудная в сильном смысле детерминированная задача построения оптимального расписания обслуживания множества требований на одном приборе с критерием минимизации суммарного взвешенного запаздывания – *single machine total weighted tardiness problem* (SMTWTP) [1]. Эта задача комбинаторной оптимизации возникает в таких областях, как планирование, распределение ресурсов, маршрутизация, транспортные перевоз-

ки и др. В задаче SMTWTP для каждого требования из множества необходимо определить время начала его обслуживания, минимизируя взвешенное запаздывание завершения обслуживания всех требований относительно заданных директивных сроков. Различные модификации метода муравьиной колонии – Ant Colony Optimization (ACO) – для решения задач этого типа были предложены в работах [2-7] и других. В настоящей работе разработан метаэвристический муравьиный алгоритм с комбинированной схемой локального поиска в двух реализациях: последовательный с одной муравьиной колонией (ACO-1) и на его основе параллельный с несколькими независимыми колониями (PACO).

Пусть $\mathrm{J} = \{J_1,\ldots,J_n\}$ – множество требований, отношение порядка на множестве не определено. Для требования $J_j, j=\overline{1,n}$ заданы время его обслуживания $p_j > 0$, весовой коэффициент $w_j > 0$, характеризующий относительную важность требования, директивный срок окончания обслуживания D_j. Требуется построить расписание π последовательного обслуживания требований множества J на одном приборе без прерываний, при котором достигается минимум неубывающей функции

$$Z(\pi) = \sum_{j=1}^{n} w_j \cdot \max\{s_j + p_j - D_j; 0\} \to \min, \qquad (1)$$

где s_j – время начала обслуживания требования J_j, при этом $s_j \geq s_l + p_l \vee s_j + p_j \leq s_l \quad \forall j, l = 1,\ldots,n,\ j \neq l$; $T_j = \max\{s_j + p_j - D_j; 0\}$ – запаздывание требования J_j при расписании π. Все требования поступают на обслуживание одновременно в момент времени $t_0 = 0$, тогда $s_j \geq 0$ $\forall j = 1,\ldots,n$.

Расписание обслуживания требований $\pi = \{\pi_1,\ldots,\pi_n\}$ однозначно задается перестановкой элементов множества J, где каждый элемент π_k в π содержит номер требования, которое обслуживается на позиции *k* в последовательности, $s_{\pi_1} = 0$, $s_{\pi_k} = s_{\pi_{k-1}} + p_{\pi_{k-1}} \forall k = 2,\ldots,n$.

ACO основывается на моделировании взаимодействия нескольких искусственных аналогов муравьев, программно представляемых в виде агентов колонии, путем применения непрямой связи – следов феромона τ_{jk}. В SMTWTP на каждой итерации агенты составляют свои решения за *n* шагов, на каждом из которых применяется правило выбора требования *j* на позицию *k* в последовательности обслуживания (в расписании). Следы феромона служат распределенной численной информацией, которая учитывается муравьями для конструирования решений задачи и которую муравьи адаптивно из-

меняют для отображения опыта, накопленного в процессе поиска решения. При этом количество феромона, оставляемое агентами, пропорционально качеству решения, составленного соответствующим агентом: чем меньше значение целевой функции (1), тем больше будет оставлено феромона.

Общая схема алгоритма муравьиной колонии:

- **Шаг 1**. Выбор условно-оптимального расписания π', $Z_{best} = Z(\pi')$.
- **Шаг 2**. Инициализация параметров алгоритма.
- **Шаг 3**. Итерационный процесс нахождения решения.
  DO WHILE (проверка условия завершения цикла)
  - построение решения π_{curr}, $Z_{curr} = Z(\pi_{curr})$;
  - улучшение решения с помощью методов локального поиска;
  - глобальное обновление следов феромона;
  - Если $Z_{curr} < Z_{best}$, то обновление π' и Z_{best}.

  END DO
- **Шаг 4**. Вывод оптимального расписания π'.

В работе для получения условно-оптимального расписания используется процедура последовательного применения методов: **EDD**-расписание [3] (расписание $\pi = (j_1, \ldots, j_n)$, при котором $D_{j_i} \le D_{j_{i+1}}$; для $D_{j_i} = D_{j_{i+1}}$ выполняется $p_{j_i} \le p_{j_{i+1}}$), схемы **NEH** [4] (эвристический алгоритм Nawaz, Enscore и Ham) и предлагаемые нами схемы, именуемые в дальнейшем **LSN-TO** и **LSN-R**, различающиеся применением в теле цикла методов **two-opt** или **Random** соответственно:

DO $iteration$=1, N_{ls}

[схема **RINS** (Random-Job-Insertion-Scheme) [4] +
схема **two-opt** или схема **Random**]

END DO

Параметр $N_{ls} \in \{1,2,3\}$ – число итераций.

В алгоритме **two-opt** на каждой итерации исследуются все возможные попарные обмены требований в текущей последовательности и идентифицируется пара, которая дает самое высокое сокращение значения целевой функции по сравнению с этой последовательностью. Если найдено улучшение, то происходит перестановка соответствующих требований, в качестве текущей принимается новая последовательность и повторяется поиск новых пар, иначе останавливается процесс поиска.

В алгоритме **Random** также рассматриваются попарные обмены требований, однако требования выбираются случайно и число итераций можно варьировать через каждые *n* итераций основного муравьиного алгоритма: $N_{rand} = n^2 \div 5n^2$.

Для инициализации начального значения следа феромона τ_{jk} используется следующий алгоритм (π' – условно-оптимальное расписание) [4]:

DO $j = 1, n$
DO $k = 1, n$

Установить $\tau_{jk} = \dfrac{1}{Z_{best} \cdot \sqrt{dist}}$, где $dist = (|\text{«позиция } j \text{ в } \pi'\text{»} - k| + 1)$

END DO
END DO

η_{jk} – эвристическая информация о том, насколько хорошим кажется постановка требования j на позицию k. Для инициализации этого параметра используется одно из расписаний: Earliest Due Date (**EDD**), Modified Due Date (**MDD**), Apparent Urgency (**AU**) [5].

В этой работе применялось правило **MDD**, предложенное в [6]: $mdd_j = \max\{C + p_j, D_j\} - C$, где C – общее время обслуживания упорядоченных требований, которые сортируются в порядке неубывания модифицированных значений директивных сроков mdd. Далее используется тот же алгоритм инициализации матрицы η_{jk}, что и для матрицы τ_{jk}.

Допустимым решением для задачи SMTWTP является последовательность π_{curr} – возможный порядок запуска требований на обслуживание. Искусственный муравей на каждом из n шагов с некоторой вероятностью q выбирает требование j для подстановки на место k в расписании по правилу [6]:

$$j = \arg\max_{j \in S} \left\{ \left(\sum_{l=1}^{k} \tau_{jl} \right)^{\alpha} \cdot \eta_{jk}^{\beta} \right\}, q_0 > q. \qquad (2)$$

Иначе j определяется по принципу «колеса рулетки» согласно распределению вероятностей P_{jk}:

$$P_{jk} = \begin{cases} \dfrac{\left(\sum_{l=1}^{k} \tau_{jl} \right)^{\alpha} \cdot \eta_{jk}^{\beta}}{\sum_{h \in \Omega} \left(\sum_{l=1}^{k} \tau_{hl} \right)^{\alpha} \cdot \eta_{hk}^{\beta}}, & j \in \Omega, \\ 0, & j \notin \Omega, \end{cases} \qquad (3)$$

где S – множество еще неупорядоченных требований; α и β – параметры, определяющие соответственно относительное влияние следа феромона и эвристической информации; $0 \le q_0 < 1$ – параметр алгоритма. Ω – множество элитных требований, которое определяется динамически в течение построения решения и содержит первые $\dim_{cand}$ еще неупорядоченных требований в порядке их следования в π' – лучшем найденном расписании к настоящему

моменту, $\dim_{cand} = |\Omega|$. Когда число неупорядоченных требований меньше $\dim_{cand}$, то рассматриваются оставшиеся.

После того, как агент построил одно из возможных решений π_{curr}, для улучшения решения применяется метод локального поиска **LSN-TO** или **LSN-R**, где π_{curr} – входящая последовательность. Локальный поиск начинает работу с некоторого начального решения и многократно пытается улучшить это решение.

В алгоритме муравьиной колонии используются два правила обновления следов феромона: глобальное и локальное. В работе [4] был предложен эффективный способ глобального обновления, который применяется после построения решения агентом и применения алгоритмов локального поиска. Обновление феромона непосредственно после того, как муравей добавил новое требование *j* к частичной последовательности на позицию *k*, является локальным. В этой работе след феромона изменяется по глобальному правилу [4] и локальному правилу $\tau_{jk} = \rho \cdot \tau_{jk} + (1-\rho) \cdot \tau_0$, τ_0 – параметр алгоритма.

В настоящей работе реализован параллельный метод PACO, в котором одновременно каждая из *p* муравьиных колоний осуществляет независимый от остальных последовательный поиск оптимального решения, используя свою локальную матрицу следа феромона. При этом отсутствует обмен информацией о следах феромона, текущих приближенных решениях в колониях и других тому подобных промежуточных данных процесса поиска решения. Однако как только одна из колоний найдет решение, об этом сообщается остальным и все колонии завершают свою работу. В программной параллельной реализации использовались коллективные функции обменов: MPI_Bcast, MPI_Allreduce, MPI_Reduce.

Для исследования эффективности предложенных методов, как и многими другими авторами, были использованы 125 тестовых задач различной сложности для размерности n = 40, n = 50, n = 100, доступных через OR-library[1]. Решения для всех задач известны, поэтому поиск продолжается до тех пор, пока не найдено это оптимальное расписание. При этом фиксируется время работы программы в секундах. Ограничение времени работы блока поиска решения колонией: $T_{lim} = 100$.

Сходимость алгоритма муравьиной колонии гарантирована, однако из-за его стохастичности время решения заранее не определено. Для получения более надежных результатов испытаний выполнялся $N_{run} = 5 \div 25$ раз пакет из 3125 решаемых тестовых задач: из OR-library взято 125 задач, для каждой из них выполнялось 25 запусков с различными последовательностями генератора случайных чисел.

[1] http://people.brunel.ac.uk/~mastjjb/jeb/info.html

В результате проведенных вычислительных экспериментов было установлено, что наиболее эффективной схемой для n = 50 и n = 100 является схема **LSN-R** при $N_{ls} = 3$ (**LS3-R**). Однако для размерности задачи n = 40 это значение должно быть другим: $N_{ls} = 2$ (**LS2-R**). Представленные ниже результаты были получены с помощью указанных методов локального поиска.

Одним из недостатков муравьиных алгоритмов является сильная зависимость от настраиваемых параметров, которые подбираются исходя из экспериментов. Для n = 40, 50 использовались параметры α = 0,2, β = 2, $\rho = q_0 =$ 0,9, $\tau_0 = 0{,}004$, $\dim_{cand} = 20$, число запусков $N_{run} = 25$. Для n = 100: α = 0,2, $\beta = 2$, $\rho = 0{,}9$, $q_0 = 0{,}5$, $\tau_0 = 0{,}01$, $\dim_{cand} = 5$, $N_{run} = 5$.

Приведем показатели эффективности поиска оптимальных решений последовательными алгоритмами ACO-1 и [5] (табл. 1). В [5] для n = 40, 50, 100 использовались параметры α = 1, β = 2, $\rho = q_0 = 0{,}9$, $\tau_0 = 0{,}004$, $\dim_{cand} = 20$ и правило инициализации матрицы $\eta_{jk} = 1/\max\{C + p_j, D_j\}$, C – общее время обслуживания упорядоченных требований. Значения времени работы программы $t_{\min}$ (минимальное), t_{avg} (среднее), $t_{\max}$ (максимальное) были найдены по всем запускам решения тестовых задач. Кроме того, для n = 100 определена величина t_σ – степень разброса времен поиска решения от среднего t_{avg}, для нахождения которой использовалась несмещенная оценка среднеквадратичного отклонения величины t_i – времени решения i-й задачи. В каждом из $N_{run} = 5$ запусков пакетов тестовых задач алгоритмом ACO-1 были найдены 100% оптимальных решений.

Таблица 1

Эффективность алгоритмов муравьиной колонии

Алгоритм, процессор	n = 40		n = 50		n = 100			
	t_{avg}	$t_{\max}$	t_{avg}	$t_{\max}$	t_σ	$t_{\min}$	t_{avg}	$t_{\max}$
ACO-1, Opteron 2,4ГГц	0,005	0,89	0,01	0,87	2,12	0,00002	0,66	55,6
[5], Pentium-III 450МГц	0,09	1,72	0,32	10,7	7,02	0,018	6,99	86,3

В [7] нами подробно изложены результаты исследования эффективности последовательного алгоритма ACO-1. Далее представлены результаты, в основном, для параллельного варианта PACO при n = 100 с параметрами из [5] и другими предложенными в настоящей работе (см. выше). Для каждого запуска пакета тестовых задач запоминалось число найденных оптимальных решений (в %). Если было найдено 100% точных решений, то такой запуск

был полностью успешным. В табл. 2 в скобках указано число успешных запусков из проведенных пяти. В этой таблице также показаны значения среднего времени t_{avg} нахождения точного решения для лучшего запуска тестового пакета. В последовательном режиме (p = 1) работы PACO (ACO-1) с параметрами из [5] не было найдено 100% решений. В лучшем из пяти запусков было найдено 91,9% решений с относительной погрешностью $\delta = 4{,}6\,\%$, которая вычислялась по формуле: $\delta = 100 \cdot |Z_{opt} - Z_{best}|/Z_{best}$, где Z_{opt} – взвешенное суммарное запаздывание для известного лучшего решения, Z_{best} – для полученного в ходе работы программы. Использование значений параметров [5] приводит к достижению уровня успешности 100% хотя бы в одном из пяти запусков только при $p > 5$.

Таблица 2

Эффективность алгоритма PACO для различного числа колоний

p		1	2	4	5	6	10
PACO	%	100(5)	100(5)	100(5)	100(5)	100(5)	100(5)
	t_{avg}	0,66	0,34	0,18	0,15	0,14	0,092
PACO с параметрами [5]	%	91,9	98,1	99,9	99,9	100(1)	100(5)
	t_{avg}	2,74	1,32	0,60	0,46	0,37	0,17

Таким образом, предложенный в настоящей работе PACO с параметрами $\alpha = 0{,}2$, $\beta = 2$, $\rho = 0{,}9$, $q_0 = 0{,}5$, $\tau_0 = 0{,}01$, $\dim_{cand} = 5$ показал свою эффективность как в последовательном способе запуска, так и в параллельном. При увеличении количества параллельных процессов уменьшается влияние значений параметров на эффективность нахождения оптимальных решений (табл. 2).

Предложенная параллельная реализация алгоритмов обеспечивает стабильность времени решения задач любой сложности. При этом не только среднее время решения тестовых задач, но и максимальное время их решения существенно уменьшается с увеличением числа задействованных параллельных процессов p (табл. 3).

Таблица 3

Максимальное время работы программы PACO

p	1	2	3	5	8	16	24	48	72
t_{max}	55,6	32,3	18,6	10,7	9,8	4,7	3,3	1,6	0,8

Об эффективности распараллеливания можно судить по коэффициенту ускорения работы программы $s_p = t_1/t_p$, где t_1 – время решения задачи без

распараллеливания, t_p – время выполнения программы на p параллельных процессах, и эффективности $e_p = s_p / p$. В табл. 4 приведены данные показатели для различного числа задействованных процессов при решении набора тестовых задач для $n = 100$.

Таблица 4

Показатели эффективности параллельной реализации

p	2	3	5	8	16	24	48	72
s_p	1,9	2,8	4,3	6,4	10	13	17	19
e_p	0,97	0,93	0,87	0,80	0,63	0,54	0,35	0,26

Расчеты проводились на узлах вычислительного кластера ВЦ ДВО РАН, оснащенных четырьмя шестиядерными процессорами AMD Opteron Istanbul 8431 (2,4 ГГц).

Предложенный в работе параллельный метаэвристический гибридный метод решения SMTWTP, в котором муравьиный алгоритм сочетается с локальным поиском, показал свою эффективность – для всех тестовых задач были получены оптимальные решения с небольшим разбросом времени решения. Данный подход к построению алгоритмов может быть использован при решении других сложных задач комбинаторной оптимизации большой размерности.

Библиографические ссылки

1. *Martin Josef Geiger*. The single machine total weighted tardiness problem – is it (for metaheuristics) a solved problem? // The VIII Metaheuristics International Conference. Hamburg, 2009.

2. *An ant* colony optimization approach for the single machine total tardiness problem / A. Bauer, B. Bullnheimer, R. F. Hartl, C. Strauss // In Proceedings of the 1999 Congress on Evolutionary Computation CEC99. Piscataway, NJ, 1999. Vol. 2.

3. *Лазарев А. А., Гафаров Е. Р.* Теория расписаний. Минимизация суммарного запаздывания для одного прибора. М.: Вычислительный центр им. А. А. Дородницына РАН, 2006.

4. *Holthaus O., Rajendran C.* A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs // Journal of the Operational Research Society. 2005. No. 56.

5. *Den Besten M., Stützle T., Dorigo M.* Ant colony optimization for the total weighted tardiness problem // Lecture Notes in Computer Science, Springer-Verlag. Berlin, 2000. Vol. 1971.

6. *Merkle D., Middendorf M.* An ant algorithm with a new pheromone evaluation rule for total tardiness problems // Lecture Notes in Computer Science, Springer-Verlag. Berlin, 2000. Vol. 1903.

7. *Долгова О. Э., Пересветов В. В.* Метод параллельных муравьиных колоний в минимизации суммарного взвешенного запаздывания для одного прибора: Препринт ВЦ ДВО РАН. Хабаровск, 2011. № 171.