ИНФОРМАТИКА

2012 **октябрь-декабрь** **№ 4**

ПО МАТЕРИАЛАМ ПЯТОЙ МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ «ТАНАЕВСКИЕ ЧТЕНИЯ»

УДК 681.32

О. Голами, Ю.Н. Сотсков

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ПОСТРОЕНИЯ РАСПИСАНИЙ ОБСЛУЖИВАНИЯ ТРЕБОВАНИЙ С РАЗЛИЧНЫМИ МАРШРУТАМИ

Задача построения оптимального расписания обслуживания m приборами n требований с различными маршрутами является NP-трудной при любом m > 2 для всех регулярных критериев, рассматриваемых в теории расписаний. Для ее решения разработаны эвристические алгоритмы для трех регулярных критериев: минимизации общего времени обслуживания заданных требований; минимизации суммарного времени обслуживания n требований и минимизации суммарного запаздывания обслуживания n требований. Экспериментальное сравнение разработанных программ с одним из наиболее эффективных эвристических алгоритмов показало их превосходство по времени реализации и достаточно близкие результаты по качеству получаемых расписаний в случае, когда число m больше числа n. Неравенство m > n выполняется, в частности, для задач, возникающих при составлении оптимальных расписаний движения поездов по одноколейным железным дорогам.

Введение

В задачах оперативно-календарного планирования, возникающих на транспорте, в промышленности, в компьютерных технологиях, при планировании сервиса, в логистике и т. п., требуется выполнить множество работ на соответствующих рабочих местах (т. е. обслужить множество требований приборами) без прерывания операций и при соблюдении заданных временных и ресурсных ограничений. Временные ограничения указывают на то, что одни операции должны быть закончены раньше, чем начаты другие. Ресурсные ограничения задают недопустимость одновременного выполнения двух и более операций по обслуживанию требований одним и тем же прибором.

В подобного рода задачах требуется построить оптимальное расписание, т. е. определить, когда должна начаться каждая из заданных непрерываемых операций, чтобы соблюдались как временные, так и ресурсные ограничения, причем не убывающая от моментов завершения обслуживания требований целевая функция принимала бы минимальное значение. Такой критерий оптимальности расписаний принято называть регулярным [1, 2]. Используя терминологию теории расписаний [1], задачу можно сформулировать следующим образом.

Множество требований $J = \{J_1, J_2, ..., J_n\}$ необходимо обслужить приборами из множества $M = \{M_1, M_2, ..., M_m\}$ так, чтобы заданная целевая функция принимала наименьшее из возможных значений. Приборы множества M попарно различны по их назначению, т. е. для каждой операции по обслуживанию каждого требования заранее известен прибор из множества M, который предназначен для выполнения этой операции. Соответственно технологический маршрут $O_i = (O_{i1}, O_{i2}, ..., O_{in_i})$ обслуживания каждого требования $J_i \in J$ заранее определен, причем для разных требований из множества J могут быть априори заданы различные технологические маршруты. Время p_{ij} выполнения операции O_{ij} по обслуживанию требования $J_i \in J$ соответствующим прибором $M_v \in M$ известно до начала составления расписания обслуживания требований из множества J.

Для многостадийных обслуживающих систем [1] чаще других регулярных критериев рассматривается критерий $C_{\max}$, при котором требуется построить оптимальное по быстродействию расписание обслуживания требований J, т. е. расписание с наименьшей продолжительно-

стью. Иными словами, необходимо минимизировать общее время $C_{\max} = \max\{C_i : J_i \in J\}$ обслуживания требований из множества J [1, 2]. Здесь и далее C_i обозначает момент завершения обслуживания требования $J_i \in J$.

В англоязычной литературе [2] сформулированную задачу принято называть задачей job-shop (цех работ) с критерием $C_{\max}$. В общепринятой трехпозиционной классификации задач теории расписаний [2] такая задача обозначается $J \| C_{\max}$.

Для решения задачи $J \| C_{\max}$ разработаны как точные, так и эвристические алгоритмы. Среди точных алгоритмов следует отметить алгоритмы типа ветвей и границ (branch-and-bound), которые позволили найти оптимальные расписания для рекордных по размерности тестовых задач $J \| C_{\max}$ с доказательством оптимальности полученных решений [3, 4]. Следует, однако, признать, что размерности задач job-shop, для которых удается найти оптимальные расписания и доказать их оптимальность, все еще не столь велики, чтобы точные алгоритмы можно было широко использовать в практике оперативно-календарного планирования.

По-видимому, в ближайшем будущем «проблема размерности» так и не будет решена, поскольку задача $J \| C_{\max}$ является NP-трудной уже при $m > 2$ [1, 2], а в случае, когда допускаются повторения приборов в маршруте обслуживания требования, задача $J \| C_{\max}$ становится NP-трудной при $n = m = 3$ [5].

Среди эвристических алгоритмов, разработанных для решения задачи $J \| C_{\max}$, отметим алгоритмы сдвига узкого места (shifting bottleneck [6, 7]), алгоритмы моделируемого отжига (simulated annealing), алгоритмы поиска с запретами (tabu search [8−10]), а также генетические (genetic) алгоритмы [11].

1. Эвристические алгоритмы для решения задачи $J \| C_{\max}$

Разработанный специально для задачи $J \| C_{\max}$ алгоритм сдвига узкого места [6, 7] считается одним из наиболее эффективных (по качеству получаемых приближенных решений) алгоритмов для эвристического решения задачи $J \| C_{\max}$. В основе этого алгоритма лежит поиск наилучшего расписания обслуживания требований одним из приборов множества $M = \{M_1, M_2, ..., M_m\}$, который определяет узкое место (bottleneck) для всего процесса обслуживания требований множества J. Для определения такого прибора вначале вычисляется время t_i, необходимое для обслуживания каждого требования $J_i \in J$ без учета ресурсных ограничений (т. е. без учета недопустимости одновременного выполнения двух и более операций по обслуживанию требований одним и тем же прибором). Обозначим $t = \max\{t_i : J_i \in J\}$.

Минимальное из возможных запаздываний относительно времени t расписания, построенного с учетом ресурсных ограничений для одного (каждого) прибора из множества M, вычисляется в результате поиска последовательности выполняемых на приборе операций, при которой уменьшается максимальное запаздывание относительно t всех обслуживаемых на этом приборе требований. Затем строится оптимальное расписание (или расписание, близкое к оптимальному) обслуживания требований прибором, определяющим узкое место. При этом решается соответствующая задача $1 | r_i, prec | L_{\max}$ минимизации максимального временного смещения $L_{\max} = \max\{C_i - d_i : J_i \in J\}$ при обслуживании частично упорядоченного множества требований J_i, готовых к обслуживанию в моменты времени $r_i \geq 0$ и для которых установлены директивные сроки d_i обслуживания требований $J_i \in J$. Здесь и далее *prec* обозначает, что на множестве требований J априори задано отношение строгого порядка, которое определяет допустимые последовательности обслуживания требований (соответствующие временные ограничения). Поскольку и задача $1 | r_i, prec | L_{\max}$ является NP-трудной, для ее решения, как правило, применяется эвристическая процедура, которая является достаточно эффективной как по

временным затратам, так и по качеству получаемого эвристического решения задачи $1|r_i, prec|L_{\max}$.

В результате решения (точного или приближенного) задачи $1|r_i, prec|L_{\max}$ получаются новые временные ограничения, которые добавляются к исходным временным ограничениям задачи $J||C_{\max}$. В результате добавления новых временных ограничений задача $J||C_{\max}$ превращается в задачу $J|prec|C_{\max}$. Затем из множества еще не рассмотренных на данный момент приборов множества $M=\{M_1,M_2,...,M_m\}$ выделяется прибор, определяющий следующее узкое место, и для него аналогично решается (точно или приближенно) соответствующая задача $1|r_i, prec|L_{\max}$.

Описанный выше процесс продолжается до тех пор, пока либо не будет решена задача $1|r_i, prec|L_{\max}$ для каждого из приборов множества $M=\{M_1,M_2,...,M_m\}$, либо максимальное запаздывание относительно последнего полученного времени t не окажется равным нулю для всех нерассмотренных приборов из множества M.

Как показали проведенные на ноутбуке вычислительные эксперименты, применение описанного выше алгоритма сдвига узкого места для решения задач $J|r_i|C_{\max}$, для которых выполняется неравенство $m>n$, требует значительных затрат процессорного времени.

Алгоритм поиска с запретами представляет собой локальный поиск, широко используемый при решении многих оптимизационных задач [8]. Алгоритм поиска с запретами осуществляет локальный поиск с памятью, представляемой в виде «запрещенного списка» шагов (tabu list), которые были сделаны на предыдущих итерациях алгоритма локального поиска и которые запрещаются либо на всех последующих итерациях, либо на некоторых итерациях локального поиска. Запрещенный шаг из tabu list может быть снова разрешен (несмотря на то, что ранее он был запрещен), если будут соблюдены определенные условия (например, если решение, полученное на ранее запрещенном шаге, окажется лучше наилучшего из решений, полученных до текущей итерации алгоритма).

В алгоритме поиска с запретами структура и размеры рассматриваемых в локальном поиске окрестностей возможных решений играют существенную роль для достижения хорошего расписания в смысле его качества (близости к оптимальному расписанию) и времени работы компьютера при реализации алгоритма поиска с запретами. Так, при выборе малых по размеру окрестностей расписание, которое сильно отличается от исходного расписания, не может быть найдено алгоритмом поиска с запретами. Наоборот, выбор больших по размеру окрестностей в алгоритме поиска с запретами может привести к большим затратам процессорного времени для того, чтобы достичь хорошее по качеству расписание [9].

Комбинация алгоритма сдвига узкого места и алгоритма поиска с запретами была предложена и исследована в [10]. Алгоритм сдвига узкого места использовался для получения начального эвристического решения задачи job-shop, а алгоритм поиска с запретами использовался для того, чтобы минимизировать общее взвешенное запаздывание в задаче job-shop.

2. Быстрый эвристический алгоритм для решения задачи $J|r_i|C_{\max}$ при $m>n$

Цель данного исследования состояла в разработке и апробации на тестовых примерах эвристического алгоритма для решения задачи $J|r_i|C_{\max}$, в которой требования $J_i \in J$ готовы к обслуживанию в заранее заданные моменты времени $r_i \geq 0$. Рассмотренная во введении и разд. 1 задача $J||C_{\max}$ является частным случаем задачи $J|r_i|C_{\max}$, поскольку в задаче $J||C_{\max}$ все требования считаются готовыми к обслуживанию одновременно в момент времени $t=0$, т. е. $r_i=0$ для всех требований $J_i \in J$.

С учетом специфики планируемого применения алгоритма для решения практических задач $J|r_i|C_{\max}$ было необходимо, чтобы алгоритм не требовал много процессорного времени при решении задач job-shop с входными данными большой размерности при выполнении неравенства $m>n$ и при условии, что каждый прибор может выполнять не более одной операции в

каждом маршруте O_i по обслуживанию требования $J_i \in J$. Задача job-shop с указанными особенностями возникает при составлении оптимального расписания движения поездов по одноколейной железной дороге [12].

В Иране и других странах Ближнего Востока сети железных дорог в основном одноколейные. Сеть железных дорог называют одноколейной, если пара соседних станций может соединяться только одним железнодорожным путем. Будем называть железнодорожной секцией (или просто «секцией») участок железнодорожного пути, соединяющий две соседние станции. В соответствующей задаче job-shop требуется найти наилучшее расписание (т. е. расписание с наименьшей продолжительностью) движения поездов среди всех расписаний, для которых в любой момент времени по железнодорожному пути (секции), связывающему две соседние станции, может двигаться не более одного поезда.

Как было впервые замечено в статье [12], задача составления оптимального по быстродействию расписания движения поездов по одноколейной железной дороге эквивалентна задаче $J \,||\, C_{\max}$. При этом поезда и секции можно рассматривать как синонимы соответственно требований $J_i \in J$ и приборов $M_v \in M$ в эквивалентной задаче job-shop. Операция O_{ij} представляется как движение поезда $J_i \in J$ по железнодорожной секции $M_v \in M$ одноколейной железной дороги (здесь прибор M_v предназначен для выполнения операции O_{ij}).

Известно, что генетические алгоритмы [11] требуют слишком много процессорного времени для получения приемлемого по качеству расписания. Использование релаксации Лагранжа задачи $J \,|\, r_i \,|\, C_{\max}$ или алгоритма моделируемого отжига позволяет лишь незначительно уменьшить процессорное время, требуемое для получения приемлемого по качеству эвристического решения задачи job-shop большой размерности. Другие известные эвристические алгоритмы либо требуют слишком много процессорного времени, либо позволяют получать расписания для задачи job-shop, которые оказываются далекими (по значению целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$) от оптимального расписания.

В процессе реализации на компьютере алгоритма сдвига узкого места [6, 7] оказалось, что требуется слишком много вычислений рекурсивных функций для получения промежуточных данных, используемых на каждой итерации алгоритма. Эти промежуточные данные включают директивные сроки обслуживания требований из множества J, время готовности к выполнению каждой операции, а также критический путь в орграфе $(Q, A, \varnothing)$ смешанного графа $G = (Q, A, E)$, представляющего исходные данные задачи job-shop [1, 3, 4]. Число итераций алгоритма сдвига узкого места асимптотически равно числу *m* обслуживающих приборов, поэтому время решения задачи job-shop растет с ростом *m*, и этот рост значителен для задач с числом приборов, существенно превышающим число обслуживаемых требований.

В смешанном графе $G = (Q, A, E)$ множество вершин Q представляет собой множество всех операций

$$Q = \{O, O_{1,1}, O_{1,2}, ..., O_{1n_1}, O_{2,1}, O_{2,2}, ..., O_{2n_2}, ..., O_{n1}, O_{n2}, ..., O_{nn_n} O_*\},$$

включающее также фиктивную начальную операцию O и фиктивную конечную операцию O_*. Множество дуг A определяет заданные временные ограничения, а множество ребер E – заданные ресурсные ограничения в задаче $J \,|\, r_i \,|\, C_{\max}$ [1, 2]. Каждой дуге $(O_{ij}, O_{uv}) \in A$ приписывается время (вес) p_{ij}, необходимое для выполнения операции $O_{ij} \in Q$ соответствующим прибором из множества M. Каждому ребру $[O_{ij}, O_{xz}] \in E$ приписывается пара времен (весов) p_{ij} и p_{xz}, необходимых для выполнения операции $O_{ij} \in Q$ и операции $O_{xz} \in Q$ соответственно. Дуге $(O_{in_i}, O_*) \in A$ приписывается время (вес) p_{in_i}, необходимое для выполнения операции $O_{in_i} \in Q$. Дуге $(O, O_{i1}) \in A$, $J_i \in J$, приписывается заданное время (вес) $r_i \geq 0$ готовности требования $J_i \in J$ к обслуживанию.

Наиболее раннее время начала (или наименьшее время готовности к выполнению) r_{ij} операции $O_{ij} \in Q$ может быть определено как наибольший суммарный вес пути из вершины $O \in Q$ в вершину O_{ij} в орграфе $(Q, A, \varnothing)$. Поскольку допустимый орграф $(Q, A, \varnothing)$ не должен содержать контуров, то все времена r_{ij} являются конечными и могут быть определены методом критического пути за линейное от суммы $|Q|+|A|$ число элементарных действий.

Наименьшее время готовности к выполнению (shortest release time SRT) операции $O_{ij} \in Q$ используется как приоритет (SRT-приоритет) в разработанном SRT-алгоритме для приближенного решения задачи $J \,|\, r_i \,|\, C_{\max}$. В отличие от алгоритма сдвига узкого места [6, 7], в котором на каждой итерации решается задача $1 \,|\, r_i, prec \,|\, L_{\max}$ с одним прибором из множества M, определяющим узкое место, на каждой итерации SRT-алгоритма решается задача по обслуживанию одного требования, которое является критическим на данной итерации алгоритма. Иными словами, для всех операций по обслуживанию критического требования определяется порядок их выполнения относительно операций по обслуживанию других требований из множества J на соответствующих приборах из множества M.

На первой итерации SRT-алгоритма критическим является требование $J_i \in J$, для которого последняя операция O_{in_i} в маршруте O_i является предпоследней операцией критического пути в орграфе $(Q, A, \varnothing)$ (или одного из критических путей, если таких путей в орграфе $(Q, A, \varnothing)$ несколько). На первой итерации SRT-алгоритма выполняются следующие два шага:

Шаг 1. Наименьшее время готовности r_{ij} для каждой операции $O_{ij} \in Q$ вычисляется согласно рекурсивной функции

$$r_{ij} = r_{i,j-1} + p_{i,j-1}.$$

При этом наименьшее время готовности начальной операции O полагается равным нулю. Для вычисления всех времен r_{ij}, $O_{ij} \in Q$, $J_i \in J$, требуется линейное время от числа операций $|Q|$.

Шаг 2. Начиная с первой операции O_{i1} в маршруте O_i требования $J_i \in J$, SRT-алгоритм определяет прибор $M_v \in M$, необходимый для выполнения операции O_{i1}, а также множество всех других операций $O(v) \subset Q$, которые выполняются прибором $M_v \in M$. Все ребра $[O_{i1}, O_{jk}] \in E$, для которых выполняется включение $O_{jk} \in Q(v)$, ориентируются согласно SRT-приоритету: если $r_{i1} \le r_{jk}$, то дуга (O_{i1}, O_{jk}) добавляется в орграфе $(Q, A, \varnothing)$; в противном случае симметрическая дуга (O_{jk}, O_{i1}) добавляется в орграфе $(Q, A, \varnothing)$. В обоих случаях ребро $[O_{i1}, O_{jk}]$ удаляется из смешанного графа G. Описанная процедура повторяется для операции O_{i2}, затем для операции O_{i3} и так далее до последней операции O_{in_i} в маршруте O_i обслуживания критического требования $J_i \in J$. В результате таких замен ребер из множества E на дуги смешанный граф $G = (Q, A, E)$ преобразуется в смешанный граф $G_i = (Q, A \cup A_i, E \setminus E_i)$. Конец шага 2.

На второй итерации SRT-алгоритма рассматривается «второе критическое» требование, т. е. требование $J_u \in J \setminus \{J_i\}$, для которого операция O_{un_u} в маршруте O_u имеет наибольшее время завершения в орграфе $(Q, A \cup A_i, \varnothing)$ среди всех требований из множества $J \setminus \{J_i\}$. На второй итерации требования $J_i \in J$ шаги 1 и 2 выполняются для требования J_u, орграфа $(Q, A \cup A_i, \varnothing)$ и смешанного графа G_i соответственно. При необходимости наименьшее время r_{ij} готовности операции $O_{ij} \in Q$ должно быть повторно рассчитано по следующей рекурсивной формуле:

$$r_{uj} := \max_{(O_{kl}, O_{uj}) \in A \cup A_i} \{r_{uj}, r_{kl} + p_{kl}\}.$$

Описанный процесс повторяется для «третьего критического» требования, затем для «четвертого критического» требования и так далее, пока все требования множества J не будут рассмотрены. В результате смешанный граф G превращается в орграф $G_d = (Q, A \cup A_d, \varnothing)$, здесь J_d – то требование из множества J, которое было рассмотрено на последней n-й итерации SRT-алгоритма.

Нетрудно убедиться в том, что использование SRT-приоритета при ориентации ребер E смешанного графа $G = (Q, A, E)$ не может привести к возникновению контура ни в одном из орграфов $(Q, A \cup A_i, \varnothing)$, …, $(Q, A \cup A_d, \varnothing)$, построенных на каждой итерации SRT-алгоритма. Полученный бесконтурный орграф G_d однозначно определяет активное (semi-active) расписание [2], которое может быть построено методом критического пути за время $O(n+|A_d|)$.

Помимо SRT-алгоритма авторами был разработан SCT-алгоритм, отличающийся от SRT-алгоритма только использованием в алгоритме (на шаге 2) другого правила приоритета операций при выборе порядка их выполнения в искомом расписании. Вместо SRT-приоритета в SCT-алго-ритме используется SCT-приоритет (от англ. Shortest Completion Time, т. е. кратчайшее время завершения). Первая итерация SCT-алгоритма начинается с операции O_{i1} в маршруте O_i критического требования $J_i \in J$. Согласно SCT-приоритету на шаге 2 выполняется следующая замена ребра на дугу. Если $c_{i1} \le c_{jk}$, то дуга (O_{i1}, O_{jk}) добавляется к орграфу $(Q, A, \varnothing)$. В противном случае симметрическая дуга (O_{jk}, O_{i1}) добавляется к орграфу $(Q, A, \varnothing)$.

Аналогично был разработан SDD-алгоритм, который отличается от SRT-алгоритма и SCT-алгоритма другим правилом приоритета операций. В SDD-алгоритме используется SDD-приоритет (от англ. Shortest Due Date, т. е. кратчайший директивный срок). Первая итерация SDD-алгоритма начинается с операции O_{i1} в маршруте O_i критического требования $J_i \in J$. Согласно SDD-приоритету если $d_{i1} \le d_{jk}$, то дуга (O_{i1}, O_{jk}) добавляется к орграфу $(Q, A, \varnothing)$. В противном случае симметрическая дуга (O_{jk}, O_{i1}) добавляется к орграфу $(Q, A, \varnothing)$.

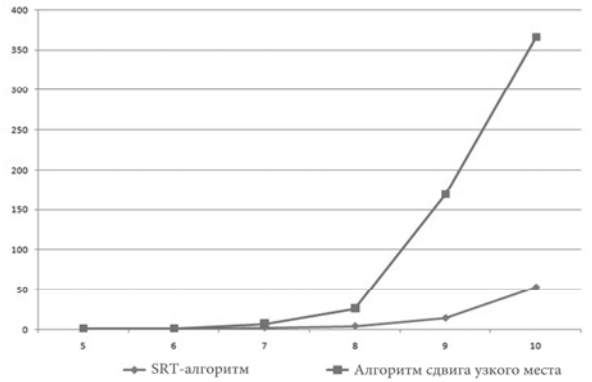
3. Результаты вычислительного эксперимента по решению случайно сгенерированных задач $J \mid r_i \mid C_{\max}$

Разработанные алгоритмы, а также алгоритм сдвига узкого места [3, 4] были закодированы на алгоритмическом языке Delphi. Для того чтобы проверить их эффективность, использовался ноутбук следующей конфигурации: Интел®, coreTM 2 Duo, CPU T6400, 2.00 GHz, 2GB Internal Memory, Windows 7, Ultimate 32 bit.

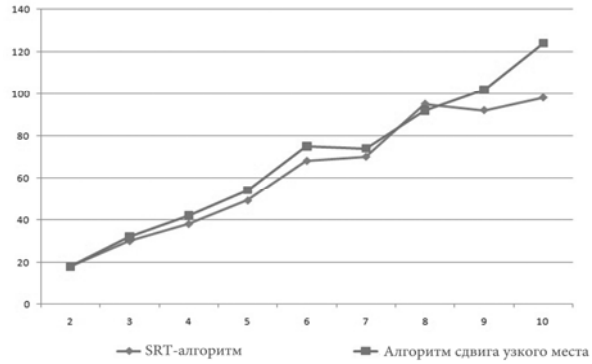
Вначале было проведено сравнение процессорного времени, которое требовалось SRT-алгоритму и алгоритму сдвига узкого места для эвристического решения одних и тех же серий случайно сгенерированных примеров задачи $J \mid\mid C_{\max}$ различной размерности $n \times m$, где $n = m$ (например, 6×6 или 10×10). Сравнивались значения критерия $C_{\max}$ для расписаний, полученных обоими алгоритмами для одинаковых входных данных. Результаты проведенного вычислительного эксперимента показаны на рис. 1. По оси абсцисс графика указаны совпадающие значения $n = m$, а по оси ординат − среднее процессорное время для серии тестовых примеров в секундах (рис. 1, *а*) и среднее значение целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ (рис. 1, *б*).

В проведенном эксперименте также оценивалось влияние добавления приборов или требований на процессорное время, необходимое для приближенного решения задачи $J \mid r_i \mid C_{\max}$ SRT-алгоритмом. Вначале решалась задача $J \mid r_i \mid C_{\max}$ с размерностью $(n = 5) \times (5 = m)$ и затем добавлялись новые требования, чтобы экспериментально оценить рост процессорного времени, необходимого для реализации SRT-алгоритма. Затем фиксировалось число требований и до-

бавлялись новые приборы в множество M для оценки влияния роста размерности задачи на процессорное время, необходимое для реализации SRT-алгоритма.



а)



б)

Рис. 1. Сравнение SRT-алгоритма и алгоритма сдвига узкого места: *а)* по процессорному времени; *б)* по значениям критерия $C_{\max}$

Как следует из эксперимента, результаты которого представлены на рис. 2, для приближенного решения задачи $J \mid r_i \mid C_{\max}$ размерности 5×20 SRT-алгоритм требует 10 с процессорного времени, а для решения задачи размерности 20×5 – 1929 с. Такое различие процессорного времени объясняется тем, что добавление требований для обслуживания в задаче job-shop влечет значительное увеличению количества вычислений приоритета в SRT-алгоритме.

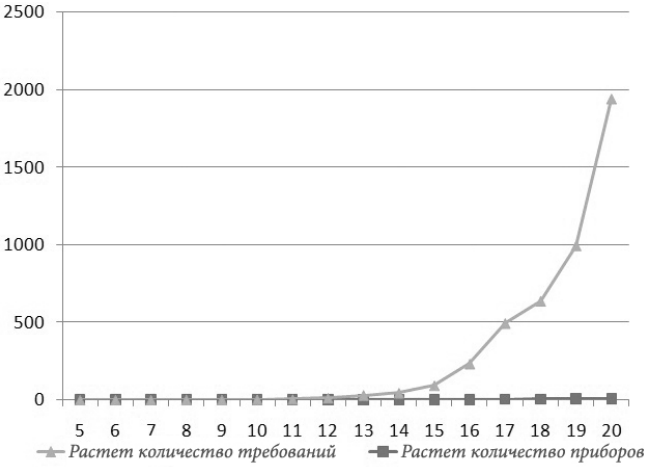


Рис. 2. Рост процессорного времени, необходимого для решения задачи $J \mid\mid C_{\max}$ SRT-алгоритмом при увеличении числа требований (или приборов)

SRT-алгоритм показывает лучшие вычислительные результаты, когда число приборов превышает число требований. Поскольку значения целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ отличаются незначительно для сравниваемых алгоритмов (алгоритма сдвига узкого места и SRT-алгоритма), можно утверждать, что SRT-алгоритм является более предпочтительным по сравнению с алгоритмом сдвига узкого места [6, 7] при решении задачи $J \,||\, C_{\max}$, для которой выполняется неравенство $m > n$.

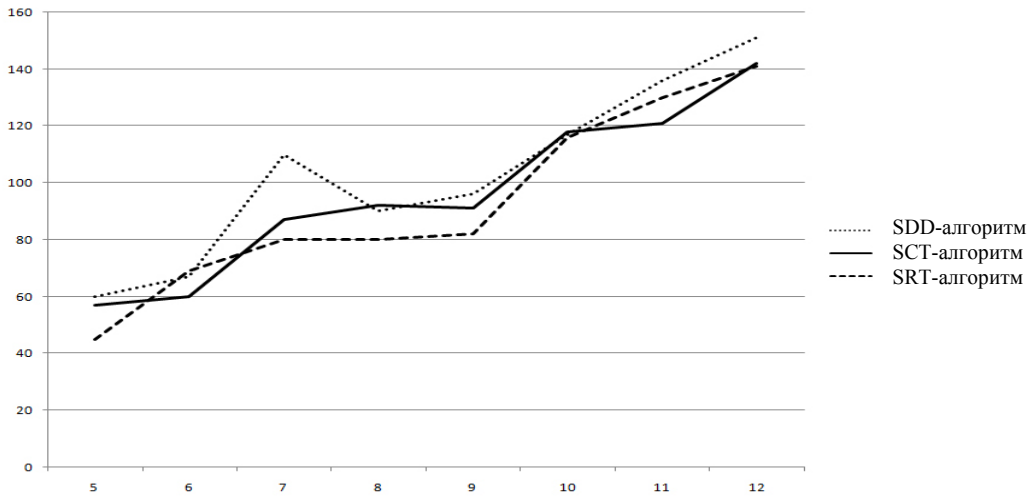


Рис. 3. Значения целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ для расписаний, полученных в результате применения SRT-, SCT- и SDD-алгоритмов

Были проведены вычислительные эксперименты и для сравнения качества расписаний (относительно критерия $C_{\max}$), полученных SRT-, SCT- и SDD-алгоритмами. Результаты этих экспериментов представлены на рис. 3. Из рисунка видно, что с помощью SRT-алгоритма строятся расписания с меньшими значениями целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ по сравнению с расписаниями, которые строит SDD-алгоритм. Качество расписаний, которые строят SRT- и SCT-алгоритмы, отличается незначительно. В большинстве проведенных экспериментов SRT-алгоритм незначительно превосходил SCT-алгоритм по значениям целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$. В остальных экспериментах, наоборот, SCT-алгоритм незначительно превосходил по качеству решений SRT-алгоритм.

4. Результаты вычислительных экспериментов по решению задач $J \,|\, r_i \,|\, \sum C_i$ и $J \,|\, r_i \,|\, \sum T_i$

При оптимизации расписания движения поездов по одноколейным железным дорогам важными критериями, наряду с критерием $C_{\max}$, являются критерий $\sum C_i = \sum_{i=1}^{n} C_i$ минимизации суммарного времени обслуживания заданных требований и критерий $\sum T_i = \sum_{i=1}^{n} T_i$ минимизации суммарного запаздывания обслуживания требований. Запаздывание T_i обслуживания требования $J_i \in J$ определяется по формуле $T_i = \max\{0, C_i - d_i\}$, где d_i – заданный директивный срок, к которому желательно обслужить требование $J_i \in J$.

В задаче $J \,|\, r_i \,|\, \sum T_i$, которая возникает при планировании движения поездов, момент времени $r_i \geq 0$ готовности требования $J_i \in J$ к обслуживанию интерпретируется как планируемое (наиболее раннее) время отправления поезда $J_i \in J$ с первой станции в его маршруте, а директивный срок d_i – как планируемое время прибытия поезда $J_i \in J$ на конечную станцию.

В связи с важностью указанных критериев для оптимального планирования железнодорожных перевозок было проведено экспериментальное сравнение значений критериев $\sum C_i$ и $\sum T_i$ для расписаний, построенных SRT-, SCT- и SDD-алгоритмами.

Результаты сравнения построенных расписаний по значениям целевой функции $\sum C_i = \sum_{i=1}^{n} C_i$ показали (рис. 4), что SCT-алгоритм существенно превосходит SDD-алгоритм по качеству приближенных решений задачи $J \mid r_i \mid \sum C_i$ и незначительно превосходит SRT-алгоритм.

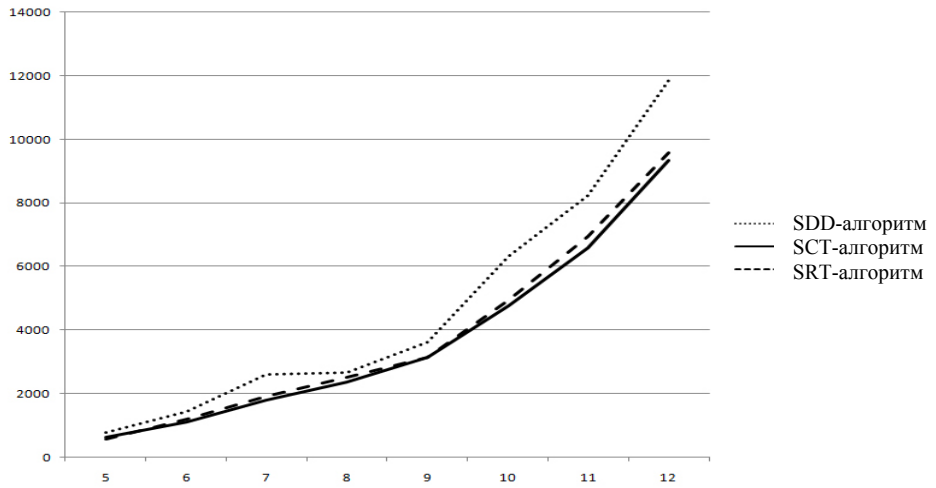


Рис. 4. Значения целевой функции $\sum C_i = \sum_{i=1}^{n} C_i$ для расписаний, полученных в результате применения SRT-, SCT- и SDD-алгоритмов

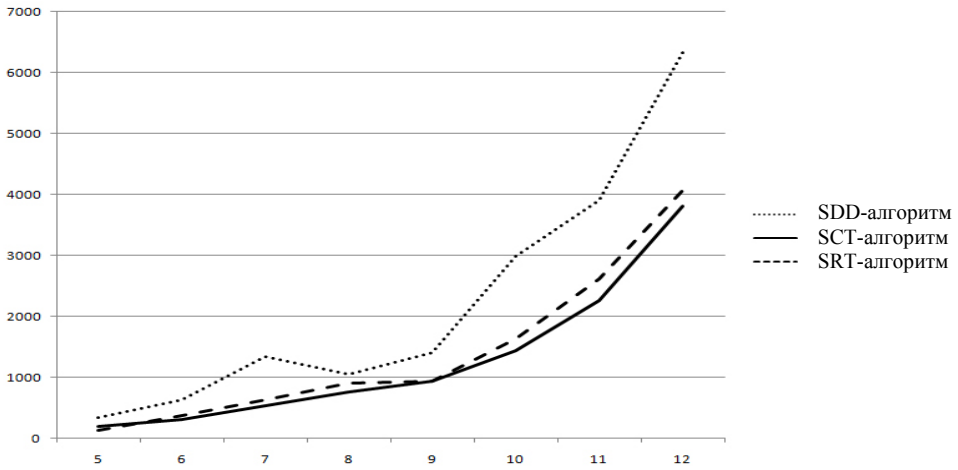


Рис. 5. Значения целевой функции $T_i = \max\{0, C_i - d_i\}$ для расписаний, полученных в результате применения SRT-, SCT- и SDD-алгоритмов

Результаты сравнения построенных расписаний по значениям целевой функции $\sum T_i = \sum_{i=1}^{n} T_i$ показали (рис. 5), что SCT-алгоритм незначительно превосходит SRT-алгоритм и существенно превосходит SDD-алгоритм по качеству получаемых приближенных решений задачи $J \mid r_i \mid \sum T_i$.

Заключение

Разработаны эвристические алгоритмы (и программы на алгоритмическом языке Delphi) для решения задачи job-shop построения оптимального расписания обслуживания m приборами n требований с различными маршрутами для следующих регулярных критериев оптимальности: минимизации общего времени обслуживания требований $C_{\max} = \max\{C_i : J_i \in J\}$; минимизации суммарного времени обслуживания требований $\sum C_i = \sum_{i=1}^{n} C_i$ и минимизации суммарного запаздывания обслуживания требований $\sum T_i = \sum_{i=1}^{n} T_i$. Исследованные критерии являются важными при решении практических задач составления оптимальных расписаний движения поездов по одноколейным железным дорогам. Особенности такого класса практических задач состоят в том, что число m обслуживающих приборов (железнодорожных станций), как правило, больше числа n обслуживаемых требований (поездов), причем каждый прибор (станция) встречается в маршруте обслуживания каждого требования (т. е. в маршруте движения поезда) не более одного раза.

Были проведены вычислительные эксперименты на ноутбуке, которые показали превосходство разработанных алгоритмов для указанного класса задач job-shop по времени реализации по сравнению с широко известным алгоритмом сдвига узкого места. По качеству получаемых расписаний были получены достаточно близкие результаты.

Статистический анализ вычислительных результатов показал, что хотя алгоритм сдвига узкого места значительно сложнее разработанных эвристик, он не имеет существенного превосходства по качеству получаемых решений по сравнению с более простыми и, следовательно, менее трудоемкими алгоритмами в классе задач job-shop при выполнении условия $m > n$. Поэтому при составлении оптимальных расписаний движения поездов по одноколейным железным дорогам целесообразно использовать алгоритмы и программы, описанные в разд. 2–4.

При экспериментальном сравнении программных реализаций разработанных алгоритмов между собой оказалось, что один из трех алгоритмов, а именно SCT-алгоритм, существенно превосходит SDD-алгоритм по качеству решений для двух из трех рассмотренных критериев (для задач $J \mid r_i \mid \sum C_i$ и $J \mid r_i \mid \sum T_i$) и незначительно превосходит SRT-алгоритм для тех же критериев. Процессорное время, которое требуется для реализации SCT-, SDD- и SRT-алгоритмов, практически одинаково.

Список литературы

1. Танаев, В.С. Теория расписаний. Многостадийные системы / В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич. – М. : Наука. Гл. ред. физ.-мат. лит, 1989. – 328 с.

2. Sequencing and scheduling: Algorithms and complexity / E.L. Lawler [et al.] // Handbooks in Operations Research and Management Science. Logistics of Production and Inventory. – North-Holland, 1993. – P. 445–522.

3. Carlier, J. An algorithm for solving the job shop problem / J. Carlier, E. Pinson // Management Science. – 1989. – Vol. 35, № 2. – P. 164–176.

4. Brucker, P. The job-shop problem and immediate selection / P. Brucker, B. Jurusch, A. Kramer // Annals of Operations Research. – 1994. – Vol. 50. – P. 73–114.

5. Sotskov, Yu.N. NP-hardness of shop-scheduling problems with three jobs / Yu.N. Sotskov, N.V. Shakhlevich // Discrete Applied Mathematics. – 1995. – Vol. 59, № 3. – P. 237–266.

6. Adams, J. The shifting bottleneck procedure for jobshop scheduling / J. Adams, E. Balas, D. Zawack // Management Science. – 1988. – Vol. 34, № 3. – P. 391–401.

7. Balas, E. Guided local search with shifting bottleneck job shop scheduling / E. Balas, A. Vazacopoulos // Management Science. – 1998. – Vol. 44, № 2. – P. 262–275.

8. Glover, F. Tabu search – part 1 / F. Glover // ORSA Journal on Computing. – 1989. – Vol. 1, № 2. – P. 190–206.

9. Dell'Amico, M. Applying tabu search to the job-shop scheduling problem / M. Dell'Amico, M. Trubian // Annals of Operations Research. – 1993. – Vol. 41. – P. 231–252.

10. Britto, R.A. Combined approach of the shifting bottleneck and tabu search heuristics for minimizing total weighted tardiness in job shop scheduling problems / R.A. Britto, G.M. Delgadillo, J.P. Villalobos // Third International Conference on Production Research (ICPR-AM06). – Brazil, 2006.

11. Werner, F. Genetic algorithms for shop scheduling problems: a survey / F. Werner. – Germany, Magdeburg, 2011. – (Preprint 31/11, FMA. Otto-von-Guericke-University).

12. Szpigel, B. Optimal train scheduling on a single line railway / B. Szpigel // Operations Research. – 1973. – Vol. 72. – P. 344–351.

Поступила 23.08.12

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: gholami_iran@yahoo.com
sotskov@newman.bas-net.by

O. Gholami, Yu.N. Sotskov

HEURISTIC ALGORITHMS FOR JOB SHOP SCHEDULING

As a job-shop problem is NP-hard, one cannot design an optimal schedule for a large job-shop problem instance in a reasonable time. Moreover, many known heuristic algorithms need a lot of CPU time to construct a sufficiently good schedule. In this paper, we present fast heuristics and computationally compare them with one of the most famous heuristics for a job-shop scheduling problem.