УДК 621.876.11 + 519.872.2

ИТЕРАЦИОННЫЙ АЛГОРИТМ ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

В.Н. НИКОНОВ, М.В. СИЛИВОНЕЦ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 1 ноября 2012

Разработан итерационный алгоритм диспетчерского управления, превосходящий стандартные алгоритмы по пропускной способности, качеству обслуживания пассажиров и энергетической эффективности.

Ключевые слова: алгоритм, диспетчерское управление, лифт.

Введение

Цель работы системы управления группой лифтов в распределении запросов на обслуживание между лифтами группы, которое минимизирует время обработки запросов. Система группового управления должна обеспечивать принятие решений о текущих действиях всех лифтов, направленных на достижение групповой цели, на основе информации о текущей ситуации и ее предыстории в реальном масштабе времени. Однако размерность такой задачи может быть слишком велика для решение в реальном масштабе времени.

Оценим трудоемкость задачи поиска оптимального распределения вызовов между лифтами в группе. Пусть некоторое высотное здание, содержащее K этажей, обслуживает группа из n $(n<K)$ лифтов. Предположим, что с каждого j-го (j=1,2,…, K) этажа здания в любой момент времени могут поступать запросы на обслуживание типа «Вверх» и «Вниз» (креме крайних этажей, где имеет смысл запрос только в одном направлении), причем запросы обоих типов могут поступать одновременно. Для формализации задачи примем:

– время перехода лифта на 1 этаж (τ) вверх или вниз одинаково для всех лифтов группы;

– время посадки/высадки пассажиров в лифт ($k\tau$) одинаково для всех лифтов группы.

Нужно разбить все множество запросов M, поступающих с различных этажей здания, на такие непересекающиеся подмножества M_i ($M=\bigcup_{i=1}^{n} M_i$, i=1, 2,…, n), чтобы максимальное время отработки i-м лифтом подмножества запросов M_i с учетом уже имеющегося внутреннего задания было минимальным. Т.е. было минимальным время

$$T=\max\{T_i; i=1,2,...,n\}, \quad (1)$$

где T – общее время обслуживания всех запросов; T_i – время обслуживания i-м лифтом подмножества запросов M_i.

Время обслуживания i–ым (i = 1, 2, ..., n) лифтом подмножества запросов M_i будет в свою очередь, складываться из времени движения i–го лифта от его текущего положения до самого «дальнего» по числу этажей запроса из подмножества M_i, а также времени посадки/высадки пассажиров во время обслуживания всех запросов из подмножества M_i и всех «внутренних» заданий на обслуживание этажей, не входящих в подмножество M_i и расположенных между этажом текущего положения i–го лифта и этажом самого «дальнего» запроса из подмножества M_i , т.е.:

$$T^i=\tau\left|N_T^i-N_P^i\right|+k\cdot\tau\cdot m_i+k\cdot\tau\cdot\nu_i, \quad (2)$$

где $m_i = |M_i|$ – число запросов в подмножестве M_i; N_T^i – номер этажа, на котором в текущий момент времени находится *i*-ый лифт; N_P^i – номер наиболее «дальнего» этажа относительно этажа N_T^i, запрос с которого включен в подмножество M_i; ν_i – число «внутренних» заданий на обслуживание этажей, расположенных между этажами N_T^i и N_P^i не входящих в подмножество M_i.

Подставляя (2) в (1), получаем:

$$T^i = \left\{ \tau \left| N_T^i - N_P^i \right| + k\tau(m_i + \nu_i); i = 1,2,...,n \right\}. \quad (3)$$

Таким образом, задача управления групповым взаимодействием лифтов сводится к задаче разбиения множества текущих запросов *M* на такие непересекающиеся подмножества M_i (*i* = 1,2,…, *n*) запросов для каждого лифта группы, в результате которого достигался бы минимум целевой функции (3).Данную задачу можно отнести к классу комбинаторных задач, для решения которых существует ряд методов. Однако все эти методы так или иначе требуют полного перебора возможных вариантов, число которых при решении задачи будет равно:

$$C = (2^n - 1)^m, \quad (4)$$

где $m = |M|$ – число запросов, *n* – число лифтов.

В максимальном случае *m* = 2(*K* – 1), где *K* – число этажей в здании. Поэтому:

$$C = (2^n - 1)^{2(K-1)}. \quad (5)$$

Рассмотрим в качестве примера здание из первого контрольного сценария работы [1]. Число этажейв рассматриваемом здании 20, а число лифтов – 4. Получаем:

$$C = (2^4 - 1)^{38} \approx 2^{152}. \quad (6)$$

Очевидно, что для большинства контроллеров такой перебор в реальном времени изменения ситуации невозможен.

Описание методики

В работе [2] предложен иной подход к подобной проблеме комбинаторного распределенияцелей, основывающийся на стратегии распределенного или коллективного принятия решения. Этот подход основывается на итерационном алгоритме оптимизации, суть которого заключается в последовательном выборе каждым объектом, входящим в группу, такой цели, которая давала бы экстремальное приращение целевого функционала при фиксированном выборе всех остальных объектов группы. Итерационный процесс оптимизации продолжается до тех пор, пока в двух последовательных циклах итерации значение целевого функционала не изменяется.

В отличие от методов решения этой задачи, основывающихся на полном переборе вариантов, данный подход позволяет существенно снизить число анализируемых вариантов, поскольку каждый лифт группы принимает решение только о своих действиях, не пытаясь при этом решать задачу оптимизации действий всех лифтов группы.

Предположим, существует некоторое начальное разбиение множества запросов *M* на подмножества M_i^j (*i* = 1,2,…, *n*, *j*= 0,1,…, где *j* – шаг итерации). Итерационный алгоритм оптимизации предполагает последовательную попытку каждого лифта группы улучшить начальное распределение для минимизации целевой функции (3). Сначала попытку делает первый лифт. Среди множества запросов *M* он выделяет подмножество M_{ig}^j (*i* = 1) допустимых для него запросов, удовлетворяющих указанным выше ограничениям, и в тоже время не включенных ранее в множество M_i^j. Далее из подмножества M_{ig}^j выделяется запрос, который включается в множества M_i^j. В результате получаем новое множество M_i^{j+1}. Заметим, если ранее данный запрос был включен в некоторое другое подмножество M_f^j, обслуживаемое *f*-ым лифтом, то соответственно это подмножество должно уменьшиться на этот запрос, т.е. *f*-ый лифт также должен

обслуживать новое множество запросов M_f^{j+1}. Далее определяется значение величины:

$$\Delta T = T_{\text{об}}{}^{j+1} - T_{\text{об}}{}^{j} = \max\{\left|N_T^i - N_P^{i,j+1}\right| + k(m_i^{j+1} + \nu_i^{j+1}); i = 1,2,...,n\} - \\ -\max\{\left|N_T^i - N_P^{i,j}\right| + k(m_i^j + \nu_i^j); i = 1,2,...,n\}, \quad (7)$$

где $T_{\text{об}}{}^{j}$ – время обслуживания, получаемое при старом распределении запросов, когда *i*-ый и *f*-ый лифты обслуживают множества запросов M_i^j и M_f^j соответственно; $T_{\text{об}}^{j+1}$ – время обслуживания, получаемое при распределении запросов, где *i*-ый и *f*-ый лифты обслуживают множества запросов M_i^{j+1} и M_f^{j+1} соответственно; N_P^{ij} – номер самого дальнего этажа в множестве M_i^j; $N_P^{i,j+1}$ – номер самого дальнего этажа в множестве M_i^{j+1}; ν_i^j – число «внутренних» заданий *i*-го лифта на обслуживание этажей, расположенных между этажами N_T^i и N_P^{ij} не входящих в множество M_i^j; ν_i^{j+1} – число «внутренних» заданий *i*-го лифта на обслуживание этажей, расположенных между этажами N_T^i и $N_P^{i,j+1}$, не входящих в множество M_i^{j+1}; $m_i^j = \left|M_i^j\right|$ и $m_i^{j+1} = \left|M_i^{j+1}\right|$ (*i*=1,2,…, *n*).

Аналогичным образом значение величины (7) определяется для всех запросов из подмножества M_{ig}^j путем их включения во множество запросов M_i^j, обслуживаемое *i*-м лифтом. Если в результате выполнения этой процедуры оказывается, что для некоторых запросов из множества M_{ig}^j выполняется условие $\Delta T < 0$, то запрос, для которого величина ΔT получается минимальной, включается во множество M_i^j, и одновременно исключается из множества M_f^j, в которое он входил ранее. Получаем новое распределение запросов, т.е. новые множества M_i^{j+1} (*i* = 1,2,…,*n*). Значение величины времени обслуживания $T_{\text{об}}^{j+1}$ для этого распределения запросов будет меньше времени $T_{\text{об}}{}^{j}$ получаемого при предыдущем распределении. Аналогичная процедура выполняется для второго лифта, т.е. формируется множество M_{2g}^{j+1} допустимых запросов, не включенных ранее в подмножество запросов M_2^{j+1}, обслуживаемых вторым лифтом. Для каждого запроса из множества M_{2g}^{j+1} определяется значением ΔT, причем в качестве начального распределения принимаем подмножества M_i^{j+1} (*i*=1,2,…, *n*), полученные в результате выполнения процедуры перераспределения запросов для первого лифта. Если для некоторых запросов из множества M_{2g}^{j+1} оказывается, что $\Delta T < 0$, то во множество M_2^{j+1} включается тот запрос, для которого величина ΔT минимальна. В то же время этот запрос исключается из множества M_l^{j+1}, в которые он входил ранее. В результате получается новое распределение запросов, т.е. новые множества M_i^{j+2} (*i*=1,2,…,*n*), для которых значение времени $T_{\text{об}}^{j+2}$ будет меньше времени обслуживания $T_{\text{об}}^{j+1}$ Процедура повторяется для всех лифтов вплоть до *n*-го, после чего итерационный цикл оптимизации повторяется вновь, начиная с первого лифта, до тех пор, пока в очередном цикле итерации не окажется, что никакое распределение запросов не приводит к уменьшению времени обслуживания. При этом полученное в предыдущем цикле распределение, т.е. множества M_i^s (*i*=1,2,…,*n*), принимается в качестве оптимального и передается на обработку лифтам. Процедура должна периодически повторяться заново с учетом всех происходящих изменений ситуации, что позволяет оперативно оптимизировать распределение с учетом этих изменений. Данная процедура является асимптотически сходящийся в смысле определения Ляпунова, причем число итерационных циклов, за которые данная процедура сходится, будет не более, чем

$$D \le \frac{\left|T_{\text{об}}^{\max} - T_{\text{об}}^{\min}\right|}{\Delta T_{\min}}, \quad (8)$$

где $T_{об}^{max}$ – максимально возможное значение времени обслуживания; $T_{об}^{min}$ – минимально возможное значение времени обслуживания; ΔT_{min} – минимально возможное изменение времени обслуживания при перераспределении запросов между лифтами.

Значения $T_{об}^{max}$, $T_{об}^{min}$ и ΔT_{min} можно оценить, исходя из следующих соображений:

1. Максимально возможное время обслуживания может быть в том случае, если все запросы последовательно будет обслуживать один единственный лифт, причем в наихудшем случае общее число обслуживаемых запросов будет 2(*K*– 1), где *K* – число этажей в здании. Поэтому:

$$T_{об}^{max} \le 2\tau(K-1)(1+k). \quad (9)$$

2. Минимально возможное время обслуживания будет достигаться в случае, если каждый из лифтов выполняет один единственный запрос, перемещаясь при этом на один этаж:

$$T_{об}^{min} \ge \tau(1+k). \quad (10)$$

3. Минимально возможное изменение времени обслуживания при перераспределении запросов достигается в случае, если один лифт «забирает» у другого один запрос. При этом общее время обслуживания в минимальном случае будет уменьшаться на время посадки-высадки пассажиров $k\tau$ или время, в течение которого лифт проезжает один этаж τ, в зависимости от того, какая из этих величин будет меньше для конкретного здания:

$$\Delta T_{min} \ge \min\{k\tau, \tau\}. \quad (11)$$

Отсюда получаем:

$$D \le \frac{(2\tau(K-1)(1+k) - \tau(1+k))}{\min\{k\tau, \tau\}} = \frac{(1+k)\cdot(2K-2)}{\min\{k,1\}}. \quad (12)$$

Описанная выше процедура оптимизации распределения запросов между лифтами обладает рядом преимуществ по сравнению со стандартными процедурами, основывающимися на полном переборе. Значительно снижется число анализируемых вариантов распределения запросов:

$$C = D\cdot n\cdot m_{ig}^{j} \le D\cdot n\cdot K = \frac{(k+1)\cdot(2K-2)}{\min\{k,1\}}\cdot n\cdot K, \quad (13)$$

где *D* – число циклов итерации;*n* – число лифтов; $m_{ig}^{j} = \left|M_{ig}^{j}\right|$ – число запросов, включенных во множество допустимых запросов для *i*-го лифта на *j*-ом шаге итерации; *K* – число этажей.

Для рассматриваемого здания *K* = 20, *n* = 4, *k* = 1, получаем: C≤5920, т.е. число анализируемых вариантов, по сравнению со случаем полного перебора снижается приблизительно в 9,6·1041 раз. Это позволяет реализовать вышеописанную процедуру в реальном времени изменения ситуации с помощью большинства распространенных вычислительных устройств. Разумеется, такое существенное сокращение анализируемого числа вариантов достигается за счет отказа о гарантии получения глобального оптимума, и достижения некоторого локального оптимума целевой функции. Однако следует отметить, что из-за быстрого и непредсказуемого заранее изменения ситуации, как например появление новых вызовов, и неизвестного заранее точного времени посадки пассажиров по уже имеющимся вызовам, глобальный оптимум может потерять актуальность. В таких условиях имеет смысл как можно чаще искать рациональный оптимум, что и обеспечивает предложенный итерационный алгоритм.

Выбор начального распределения. Значительное влияние на результаты итерационного поиска имеет выбор начального распределения вызовов по лифтам группы. Для наглядности этого влияния в таблице приводится сравнение разработанного итерационного алгоритма с различными вариантами начального распределения вызовов: кольцевой алгоритм, алгоритм зонирования, распределение вызова ближайшему лифту, распределение вызова ближайшему лифту с учетом направления движения кабины, алгоритм приоритета времени вызова. В табл. 1 приведены результаты начальных распределений до обработки итерационным алгоритмом и после последнего цикла итерации для потока вниз и смешанного потока. Межэтажный поток, как правило, имеет не высокую интенсивность и не является определяющим при выборе варианта алгоритма. Поток вверх достаточно сложен для группы лифтов и требует специальных оптимизаций. Как видно из таблицы, наилучшее результаты продемонстрировало начальное распределение вызовов

ближайшему попутному лифту.

Таблица 1. **Результаты моделирования с разными видами начального распределения**

Алгоритм		Начальное распределение				После финальной итерации			
Тип потока		Вниз		Смешанный		Вниз		Смешанный	
Интенсивность потока		6%	10%	6%	10%	6%	10%	6%	10%
Кольцевой алгоритм	$t_{ож}$	30	49	23	33	38	55	27	34
	$t_{п}$	46	54	46	55	**44**	53	45	55
	%>мин	18	41	13	25	29	41	19	23
	$P_{\text{на пасс}}$	8,42	4,12	11,2	7,27	7,64	3,31	10,7	7,19
Алгоритм зонирования	$t_{ож}$	37	67	29	50	33	56	24	36
	$t_{п}$	45	49	45	54	46	**52**	47	55
	%>мин	27	42	20	34	22	32	11	24
	$P_{\text{на пасс}}$	7,06	2,7	11,1	7,3	6,41	2,34	11,2	6,55
Вызов ближайшему лифту	$t_{ож}$	36	48	13	22	46	66	10	19
	$t_{п}$	48	52	45	55	49	57	**42**	56
	%>мин	31	39	5	12	33	44	**1**	11
	$P_{\text{на пасс}}$	3,81	1,73	8,34	5,46	**2,48**	**1,08**	8,62	**5,26**
Вызов ближайшему попутному лифту	$t_{ож}$	24	31	19	34	**22**	**35**	**9**	**17**
	$t_{п}$	46	52	47	58	47	**52**	**42**	55
	%>мин	12	22	11	22	**8**	**25**	2	**6**
	$P_{\text{на пасс}}$	4,28	1,81	7,8	5,18	3,9	1,6	9,15	5,86

Выбор критериев оптимизации. Значительное влияние на результаты работы итерационного алгоритма оказывает подбор критериев пошаговой оптимизации. Определение критериев оптимизации производилось, исходя из следующих соображений.

1. Соотношения (7) описывают оптимизацию по общему времени выполнения всех вызовов лифтами, т.е. перестановка вызовов между 2-мя лифтами на текущем шаге итерации осуществляется лишь в том случае, если такая перестановка улучшает общее время выполнения всех вызовов всей группой лифтов. При оценке времени выполнения вызовов используется ряд усредненных значений, которые на практике могут отличаться от расчетных, как, например, время посадки пассажиров в лифт. В результате глобальный прогноз о времени выполнения всех вызовов имеет ряд погрешностей, которые могут влиять на результат работы алгоритма. Рассмотрим перестановки для менее масштабного прогнозирования ситуации:

– перестановка, обеспечивающая максимальную разницу между сокращением времени выполнения вызовов 1-го лифта и увеличением времени выполнения вызовов 2-го лифта;

– перестановка вызовов, которая обеспечивает максимальное уменьшение времени выполнения одного из вызовов:

$$\Delta T = \max\{\Delta T_i - \Delta T_f\} = \max\{T_f^j - T_f^{j+1} - (T_i^{j+1} - T_i^j)\},\ f = 1,2,...,n, \qquad (14)$$

где T_i^j и $T_f{}^j$ – время обслуживания запросов M_i^j и M_f^j лифтами *i* и *f* соответственно при старом распределении запросов; T_i^{j+1} и $T_f{}^{j+1}$ – время обслуживания запросов M_i^{j+1} и M_f^{j+1} лифтами *i* и *f* соответственно при новом распределении запросов;

– перестановка, обеспечивающая минимальное время выполнения одного из вызовов:

$$\Delta T = \max\{T_i^k - T_f^k\}, i = 1,2,...,n,\ f = 1,2,...,n, k = 1,2,...,K, \qquad (15)$$

где T_i^k и $T_f{}^k$ – время выполнение *k*-го вызова лифтами *i* и *f* соответственно, с учетом очереди запросов этих лифтов, в которой могут быть более приоритетные вызовы, чем вызов *k*.

2. Рассмотрим целесообразностьсортировки списка лифтов на каждом шаге итерации, что должно приводить к передаче вызовов от наиболее загруженных лифтов к наименее загруженным. При моделировании были рассмотрены 6 вариантов итерационного алгоритма, образованные 3-мя видами условий перестановки в версиях с сортировкой лифтов и без сортировки. В табл. 2 приведены результаты работы перечисленных выше вариантов итерационного алгоритма для потока вниз и смешанного потока.

Таблица 2. **Результаты моделирования с разными критериями итерационной оптимизации**

Порядок обхода лифтов		Без сортировки				С сортировкой			
Тип потока		Вниз		Смешанный		Вниз		Смешанный	
Интенсивность потока		6%	10%	6%	10%	6%	10%	6%	10%
По времени конкретного вызова	$t_{ож}$	22	33	11	18	**15**	**25**	**8**	**16**
	$t_{п}$	47	54	43	55	45	53	**42**	**54**
	%>мин	8	22	1	8	**3**	**14**	**0**	7
	$P_{на\ пасс}$	4,47	2,03	9,22	6,07	5,93	2,55	10,2	6,63
По изменению времени работы лифта	$t_{ож}$	37	59	13	23	25	32	9	**16**
	$t_{п}$	51	55	47	59	49	51	**42**	**54**
	%>мин	26	39	3	13	14	21	2	7
	$P_{на\ пасс}$	**2,9**	**1,27**	8,03	5,26	3,13	1,71	8,93	5,55
По общему времени работы группы	$t_{ож}$	24	35	**8**	17	24	37	9	17
	$t_{п}$	47	52	43	**54**	**48**	**54**	43	55
	%>мин	13	23	**0**	**6**	13	24	1	8
	$P_{на\ пасс}$	3,86	1,61	8,98	5,82	3,63	1,57	8,75	5,65

3. Оптимизация по общему времени работы группы без сортировки и с сортировкой показали близкие результаты. В остальных случаях алгоритмы с сортировкой показывали значительное превосходство над вариантами без сортировки.

4. Оптимизация конкретного вызова с сортировкой (далее – быстрый итерационный алгоритм) по времени выполнения значительно превзошла остальные варианты оптимизаций.

5. Оптимизация по максимальному изменению времени работы лифта с сортировкой (далее – экономичный итерационный алгоритм) является наиболее эффективной с точки зрения рациональности расхода электроэнергии.

Оптимизация итерационного алгоритма в режиме интенсивного потока вверх. Рассмотрим два варианта оптимизаций для интенсивного потока вверх: все свободные лифты отправляются на первый этаж для минимизации времени ожидания пребывающих в холл пассажиров; ближайший к первому этажу свободный лифт отправляется на первый этаж.

Таблица 3. **Результаты моделирования для разного вида оптимизаций потока вверх**

Тип потока		Вверх		Вниз		Смешанный	
Интенсивность потока		6%	10%	6%	10%	6%	10%
Без оптимизации	$t_{ож}$	18	-	**15**	25	**8**	16
	$t_{п}$	56	-	**45**	**53**	42	54
	%>мин	7	-	**3**	14	**0**	7
	$P_{на\ пасс}$	**15,5**	-	**5,93**	2,55	**10,2**	**6,63**
Все свободные лифты вниз	$t_{ож}$	**6**	26	19	26	12	16
	$t_{п}$	**46**	**62**	48	54	42	53
	%>мин	**0**	12	6	16	4	6
	$P_{на\ пасс}$	22,8	15,1	8,2	2,92	14,7	7,2
Ближайший из свободных лифтов вниз	$t_{ож}$	**6**	**21**	19	25	12	18
	$t_{п}$	47	**62**	47	54	43	55
	%>мин	**0**	11	7	13	3	7
	$P_{на\ пасс}$	20,5	14,8	8,1	2,83	13,7	7,22
Оптимизация включается автоматически	$t_{ож}$	7	22	**15**	**22**	10	**15**
	$t_{п}$	48	**62**	**45**	**53**	**41**	**53**
	%>мин	**0**	**10**	**3**	**10**	1	**5**
	$P_{на\ пасс}$	19,8	**14,7**	5,95	**2,47**	12,5	7,06

Из табл. 3 видно, что оптимизация значительно улучшает производительность группы лифтов для интенсивного потока вверх. Перевес в экономичности оказался на стороне варианта с отправкой одного лифта. Использование оптимизации для потока вверх ухудшает показатели алгоритма в других режимах. Алгоритм не имеет информации о количестве ожидающих внизу пассажиров – при любом их количестве будет зарегистрирован лишь один вызов вверх с первого этажа. Для определения характера пассажиропотока были использованы и эмпирически проверены следующие критерии: время, в течение которого вызов вверх активен в течение последних 5 минут, говорит о наличии постоянно пребывающих в холл пассажиров (t); доля заре-

гистрированных вызовов вверх с первого этажа в общем количестве зарегистрированных в течение последних пяти минут вызовов позволяет отделить интенсивный поток вверх от интенсивного смешанного потока (*a*).

Критерии включения оптимизации потока вверх строже, чем критерии отключения во избежание частого переключения режима работы при граничных значениях. Критерием включения оптимизации были эмпирически подобраны $t = 0{,}66$ и $a = 0{,}5$ или $t = 0{,}95$ при любом значении a, отключения – $t = 0{,}5$ и $a = 0{,}3$. Как видно из таблицы, автоматическое отключение оптимизации потока вверх оказывается достаточно эффективным.

Сравнительный анализ алгоритмов

Для сравнения качественных показателей разработанных итерационных алгоритмов были взяты алгоритм трех переходов (как лучший из стандартных алгоритмов [1]) и кольцевой алгоритм (широко распространен). Результаты прогонов для интенсивностей пассажиропотока 6, 8 и 10% в течение 10 мин от общего количества людей в здании сведены в таблицу.

Таблица 4. **Результаты моделирования для 11-часового прогона**

Интенсивность потока	6%				8%				10%			
Алгоритм	$t_{ож}$	$t_{п}$	%>мин	$P_{на\ пасс}$	$t_{ож}$	$t_{п}$	%>мин	$P_{на\ пасс}$	$t_{ож}$	$t_{п}$	%>мин	$P_{на\ пасс}$
Кольцевой	19	**42**	7	14,5	48	**46**	17	12,1	-	-	-	-
Трехпереходов	18	**42**	5	14,9	24	**46**	10	12,4	-	-	-	-
Быстрый итерационный	**10**	**42**	**1**	13,6	**12**	**46**	**2**	11,4	**20**	**51**	**8**	9,58
Экономичный итерационный	14	43	5	**12,4**	17	47	7	**10,5**	23	52	12	**9,11**

Наблюдается значительное превосходство итерационных алгоритмов над стандартными алгоритмами по экономичности (6–25%), которое увеличивается с ростом интенсивности пассажиропотока. В режиме с низкой интенсивностью экономичный итерационный алгоритм, на 6% уступает алгоритму приоритета вызова по времени обслуживания пассажиров, однако расходует при этом на 25% меньше электроэнергии. Итерационные алгоритмы демонстрируют высокую устойчивость к росту интенсивности пассажиропотока. Быстрый итерационный алгоритм, превосходит стандартные алгоритмы по производительности на 4–39%.

Заключение

Разработан эффективный итерационный алгоритм диспетчерского управления. Высокие показатели алгоритма в сравнении с наиболее распространенными алгоритмами по скорости обслуживания пассажиров и экономичности подтверждены имитационным моделированием.

AN ITERATIVE ALGORITHM OF DISPATCHING CONTROL

V.N. NIKONOV, M.V. SILIVONETS

Abstract

An iterative algorithm of dispatching control, exceeding standart algorithms in terms of throughput, passengers service quality and energy efficiency is created.

Список литературы

1. *Марков А.В., Никонов В.Н., Кузнецов В.П.* // Докл. БГУИР. 2009. № 4 (42). С.78–86.
2. *Каляев И.А.* // Искусственный интеллект. 2001. № 3.