Воронин Егор Ильич
E-mail: egorvoronin04@rambler.ru
Кафедра систем автоматизированного проектирования; аспирант.

Lebedev Boris Konstantinovich
Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education "Southern Federal University".
E-mail: lbk@tsure.ru.
44, Nekrasovsky, Taganrog, 347928, Russia.
Phone: +78634371743.
The Department of Computer Aided Design; Professor.

Voronin Egor Iljich
E-mail: egorvoronin04@rambler.ru.
The Department of Computer Aided Design; Postgraduate Student.

УДК 004.032.24

А.А. Аль-Хулайди, Ю.О. Чернышев

ЭКСПЕРИМЕНТАЛЬНАЯ И ТЕОРЕТИЧЕСКАЯ ОЦЕНКИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ НАХОЖДЕНИЯ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА НА КЛАСТЕРНЫХ СИСТЕМАХ*

Описывается параллельный алгоритм, разработанный для нахождения минимального отстовного дерева, на основе алгоритма Борувки и Оценки вычислительной сложности предложенного параллельного алгоритма . Были произведены вычислительные эксперименты для оценки эффективности параллельного алгоритма Прима и разработанного параллельного алгоритма на основе метода Борувки. Проведена сравнительная характеристика экспериментальных и теоретических оценок параллельных алгоритмов поиска минимального остовного дерева на кластерных системах и многопроцессорной среде. Предложенные параллельные алгоритмы позволяют достигать значительной величины ускорения, в том числе и при использовании нескольких тысяч процессоров. Сравнительная характеристика экспериментальных и теоретических оценок параллельных алгоритмов нахождения минимального остовного дерева, показала достаточную эффективность разработанного параллельного алгоритма на основе метода Борувки для нахождения минимального остовного дерева. Данный алгоритм может, использован для разработки параллельной программы для кластера локальной сети, либо многопроцессорной вычислительной системы.

Метод Борувки; параллельное программирование; остовного дерева.

A.A. Al-Khulaidi, Y.O. Chernyshev

EXPERIMENTAL AND THEORETICAL EVALUATION OF PARALLEL ALGORITHM FOR FINDING MINIMAL SPANNING TREE ON CLUSTER SYSTEMS

This paper describes a parallel algorithm designed for finding the minimum spanning tree-based algorithm Boruvki Ratings and computational complexity of the proposed parallel algorithm. Computational experiments were carried out to assess the effectiveness of the parallel algorithm and Prim's developed a parallel algorithm based on the method Boruvki. The comparative characteristics of the experimental and theoretical evaluations of parallel algorithms for finding a minimum spanning tree for cluster systems and a multiprocessor environment. The proposed parallel algorithms can achieve a significant amount of acceleration, including the use of several thousand processors. Comparative characteristics of the experimental and theoretical evaluations of parallel algorithms

* Работа выполнена при частичной поддержке РФФИ (проект № 10-01-00481-а).

for finding the minimum spanning tree showed sufficient effectiveness of the developed parallel algorithm based on the method for finding Boruvki minimum spanning tree. This algorithm can be used to develop a parallel program for a cluster network or a multiprocessor system.

Method Boruvki; parallel programming; spanning tree.

Введение. Рассматриваемая задача может найти применение в различных областях:

1. Разработка сетей. В данном случае решается задача о соединение *n* городов в с минимальной суммарной стоимостью соединений.
2. Производство печатных плат. По аналогии с сетью, необходимо соединить *n* контактов проводниками с минимальной суммарной стоимостью.
3. Минимальное остовное дерево может использоваться для визуализации многоаспектных, многомерных данных, например, для отображения их взаимосвязи.

Разработка параллельных алгоритмов нахождения минимального остовного дерева и их теоретических оценок, для использования в многопроцессорной среде и в кластерных системах, позволяют достигать значительной величины ускорения, в том числе и при использовании нескольких тысяч процессоров. В работе [1] приведены теоретические и экспериментальные данные, полученные при выполнении параллельного алгоритма нахождения минимального остовного дерева на основе метода Прима на кластере, однако отсутствует сколько-нибудь подробное описание параллельного алгоритма нахождения минимального остовного дерева. В данной статье подробно описан параллельный алгоритм нахождения минимального остовного дерева на основе метода Борувки и его теоретическая оценка по сравнение с параллельным алгоритмом Прима, приводятся данные о его производительности, полученные экспериментальным путём.

Постановка задачи. В качестве примера на рис. 1. приведен неориентированный взвешенный граф G,содержащий 8 вершин и 12 ребер. После точки указан целочисленный все ребра.

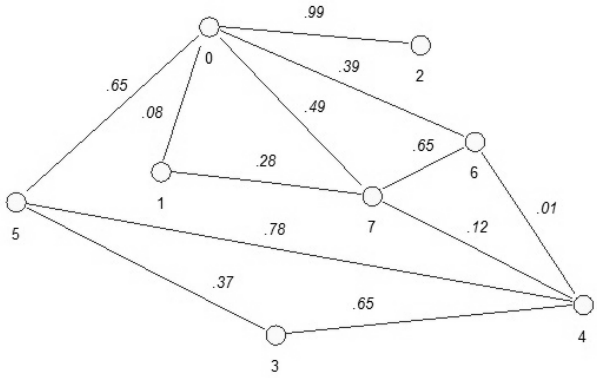

Рис. 1. Взвешенный граф

Последовательный алгоритм Борувки для нахождения минимального остовного дерева приведен в работе [2].

Разработка модификации метода Борувки для параллельных систем. Параллельный алгоритм для нахождения минимального остовного дерева состоит из следующих шагов [3].

Шаг 1 (выбор самого лёгкого): В списке ребер каждой вершины производится поиск самого легкого инцидентного ребра. На рис. 2 показан результат выполнения шага 1 при первой итерации для графа G (рис. 1). Получено 3 дерева с корнями 0,3 и 4 соответственно связные списки выглядят, как показано на рис. 3.

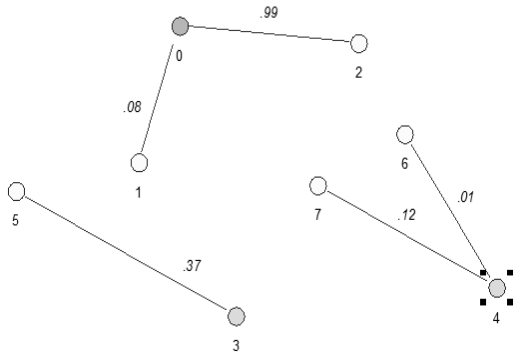

Рис. 2. Результат поиска самого легкого ребра

Шаг 2 (поиск корня): Каждая вершина ищет корень дерева, которому она принадлежит, используя алгоритм подмены указателя. Сначала корень каждого компонента устанавливает свой указатель на себя. Каждая прочая вершина, первоначально указывает на другую конечную точку самого легкого инцидентного ему ребра. Указатели вершин далее обновляются повторяющимися подменами указателя, так, что за одну замену указателя, вершина обновляет свой указатель на равный своего родителя.

Шаг 3 (переименование вершин): Каждый процессор, находит новое имя для всех вершин, находящихся в его списке ребер. Два сообщения – для запроса и ответа – требуются для получения информации от другого процессора, который ей обладает.

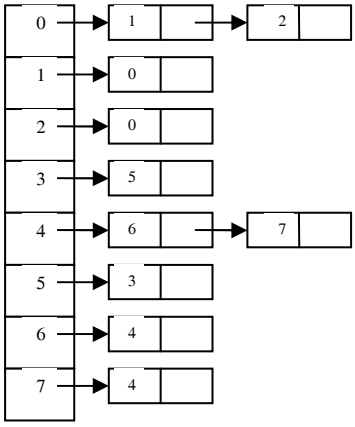

Рис. 3. Связный список графа

Шаг 4 (слияние): Ребра всех вершин компонента, посылаются процессору, который имеет список ребер корня. Список ребер далее сливается процессором.

Шаг 5 (очистка): Каждый процессор, выполняет последовательный алгоритм на своем собственном списке ребер.

В нашей реализации алгоритма подмены указателя на шаге 2 процессоры синхронизируются на каждой итерации цикла.

Рассмотрим шаг 2 более подробно. Время выполнения параллельного алгоритма на шаге 2 значительно больше, чем в последовательной версии.

1. Предлагается на шаге 2, использовать пакетную передачу, независимую от количества обновляемых указателей. При условии n<<(n/p), где n – число вершин, а p – количество процессоров, это значительно повысит производительность.
2. Новый рандомизированный алгоритм замены указателя.

Детерминистический и рандомизированный алгоритмы, требующие только линейной работы, хорошо известны. Однако они не вполне соответствуют нашей задаче по следующим соображениям: поскольку они предназначены для списков, они требуют соответствующей (линейной) формы входных данных, а не нашей древесной формы с корнем. Путем “линеаризации” дерева можно было бы использовать эти алгоритмы. Но лучше этого не делать, поскольку путь от вершины до корня в компоненте может быть значительно короче в дереве, чем в линейном списке.

Предложена новая схема замены указателей, которая далее будет называться “алгоритмом особых вершин”. Эта рандомизированная схема может быть применена к деревьям и спискам, и требует только ожидаемой линейной работы. В первом приближении, в нашем алгоритме каждый компонент обрабатывается следующим образом. Случайным образом выбирается подмножество вершин, называемых “особыми”. Назовем это подмножество SV. Каждая вершина в S-SV, выполняет простой алгоритм замены указателя, пока она не укажет на “особую” вершину. В этой точке, все вершины, кроме особых, отбрасываются и “особые” вершины повторяют тот же алгоритм рекурсивно (ещё раз с каждой вершиной, случайно выбираемой, как “особая” на следующей итерации). Как только все “особые” вершины указывают на корень, оставшиеся вершины обновляют свои указатели за один шаг так, что они тоже указывают на корень.

Оценки вычислительной сложности параллельного алгоритма Борувки. Общий анализ вычислительной сложности параллельного алгоритма Борувки для нахождения минимального остовного дерева дает идеальные показатели эффективности параллельных вычислений:

$$S_p=\frac{n^3}{(n/p)^3}=p \text{ и } E_p=\frac{n^3}{(n/p)^3 p}=1,$$

где S_p, E_p – соответственно показатели ускорения и эффективности параллельного алгоритма; n – число вершин в графе G; P – число процессоров.

Для уточнения полученных соотношений введем, в представленные выражения, время выполнения базовой операции выбора минимального значения и, учтем затраты на выполнение операций передачи данных между процессорами. Коммуникационная операция, выполняемая на каждой итерации алгоритма Борувки, состоит в передаче одной из строк матрицы A всем процессорам вычислительной системы. Как показано выше, выполнение такой операции может быть выполнено за $\log_2 p$ шагов. С учетом количества итераций алгоритма Борувки оценка длительности выполнения передачи может быть определена при помощи следующего выражения:

$$T_p(comm)=n[\log_2 p](r+vn/q),$$

где r – латентность сети передачи данных, q – пропускная способность сети, а v – размер элемента матрицы в байтах.

С учетом полученных соотношений, общее время выполнения параллельного алгоритма Борувки может быть определено следующим образом:

$$T_p=n^2\,[n/p]*\tau+\ n[log_2p](r+vn/q),$$

где τ – время выполнения выбора минимального значения элементарной операции.

Результаты вычислительных экспериментов. Были произведены вычислительные эксперименты для оценки эффективности параллельного алгоритма Прима и разработанного параллельного алгоритма на основе метода Борувки.

Эффективность параллельных алгоритмов проверялись путём выполнения их на кластере следующей конфигурации:

- 4 вычислительных узла (Intel Pentium 4 2,4 ГГц), один узел с двумя процессорами, второй и третий узлы с четырьмя процессорами, четвертый узел с восьми процессорами;
- управляющий узел (Intel Pentium 4 2,4 ГГц).

Узлы объединены между собой сетью Infiniband (пропускная способность 4 Гбит/с).

Экспериментальные исследования параллельного алгоритма Прима. Результаты вычислительных экспериментов по исследованию параллельного алгоритма Прима представлены в табл. 1, которую иллюстрирует рис. 4.

Таблица 1

Результаты экспериментального исследования параллельного алгоритма Прима

Количество вершин, n	Последовательный алгоритм, с	Параллельный алгоритм Прима					
		2 процессора, p		4 процессора, p		8 процессоров, p	
		Время, с	Ускор., с	Время, с	Ускор., с	Время, с	Ускор., с
1000	0,0435	0,2476	0,1757	0,9320	0,0467	1,5735	0,0277
2000	0,2079	0,6837	0,3041	1,7999	0,1155	2,1591	0,0963
3000	0,4849	1,4034	0,3455	2,2136	0,2191	3,1953	0,1518
4000	0,8729	1,9455	0,6220	3,3237	0,2626	5,4309	0,1607
5000	1,4324	2,6647	0,7363	2,9331	0,4884	4,1189	0,3478
6000	2,1889	2,8999	0,8214	4,2911	0,5101	7,7373	0,2829
7000	3,0424	3,2364	1,0491	6,3273	0,4808	8,8255	0,3447
8000	4,1497	4,4621	1,2822	6,9931	0,5934	10,3898	0,3994
9000	5,6218	5,8340	1,2599	7,4747	0,7521	10,7636	0,5223
10000	7,5116	6,9902	1,2915	8,5968	0,8738	14,0951	0,5329

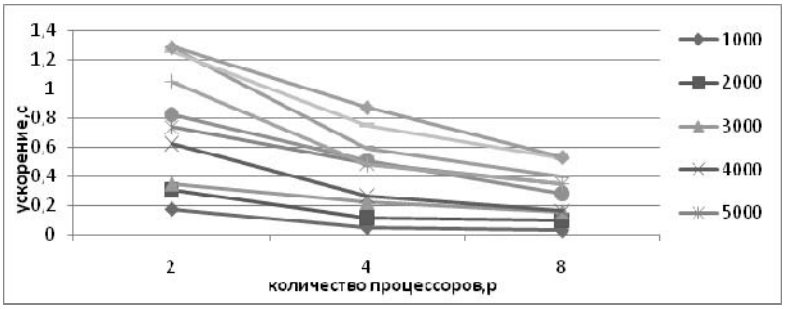

Рис. 4. Графики зависимости ускорения параллельного алгоритма Прима от числа используемых процессоров при различном количестве вершин в модели

Результаты экспериментального T^* и теоретического T исследования параллельного алгоритма Прима приведены в табл. 2.

Таблица 2

Результаты экспериментального и теоретического исследования параллельного алгоритма Прима

Количество вершин, n	Параллельный алгоритм Прима					
	2 процессора,p		4 процессора,p		8 процессоров,p	
	T_2, с	T^*_2, с	T_4, с	T^*_4, с	T_8, с	T^*_8, с
1000	0,4054	0,2476	0,8040	0,9320	1,2048	1,5735
2000	0,8203	0,6837	1,6128	1,7999	2,4120	2,1591
3000	1,2447	1,4034	2,4264	2,2136	3,6216	3,1953
4000	1,6786	1,9455	3,2447	3,3237	4,8335	5,4309
5000	2,1220	2,6647	4,0678	2,9331	6,0479	4,1189
6000	2,5750	2,8999	4,8957	4,2911	7,2646	7,7373
7000	3,0375	3,2364	5,7283	6,3273	8,4837	8,8255
8000	3,5095	4,4621	6,5656	6,9931	9,7052	10,3898
9000	3,9911	5,8340	7,4078	7,4747	10,9290	10,7636
10000	4,4821	6,9902	8,2546	8,5968	12,1552	14,0951

Результаты сравнения экспериментальной и теоретической оценки времени работы параллельного алгоритма Прима в зависимости от количество вершин в модели при использовании 2 процессоров представлены на рис. 5. Теоретическая оценка проводилась по формуле:

$$T= n^2[n/p]*\tau+n(r*log_2p+3v(p-1)/q+ log_2p(r+v/q)).$$

Как видно из табл. 2. и рис. 5, теоретические оценки определяют время выполнения алгоритма Прима с достаточно высокой погрешностью.

Экспериментальные исследования разработанного параллельного алгоритма Борувки. Результаты вычислительных экспериментов по исследованию параллельного алгоритма Борувки представлены в табл. 3, которую иллюстрирует рис. 6.

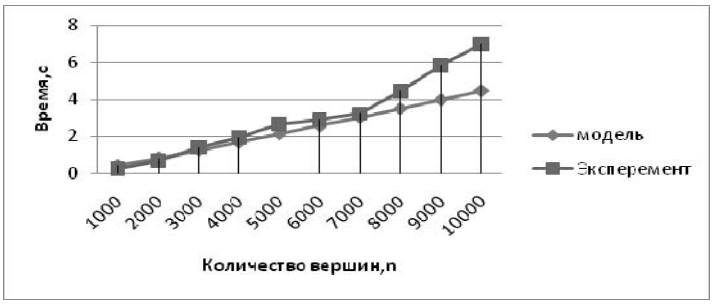

Рис. 5. Графики экспериментально установленного времени работы параллельного алгоритма Прима и теоретической оценки в зависимости от количества вершин в модели при использовании 2 процессоров

Таблица 3

Результаты вычислительных экспериментов параллельного алгоритма Борувки

Количество вершин, n	Последовательный алгоритм, с	Параллельный алгоритм Борувки					
		2 процессора		4 процессора		8 процессоров	
		Время, с	Ускор., с	Время, с	Ускор., с	Время, с	Ускор., с
1000	0,0435	0,2456	0,17711	0,832	0,05228	1,4635	0,16781
2000	0,2079	0,6821	0,30479	1,6421	0,12660	2,0499	0,33274
3000	0,4849	1,4	0,34635	2,013	0,24088	3,9992	0,35007

Окончание табл. 3

4000	0,8729	1,8123	0,48165	3,1797	0,27452	5,1518	0,35177
5000	1,4324	2,423	0,59116	2,653	0,53991	4,989	0,48566
6000	2,1889	2,658	0,82351	4,001	0,54708	7,0784	0,37550
7000	3,0424	2,9212	1,04148	6,013	0,50597	8,8255	0,33099
8000	4,1497	3,71	1,11851	6,8731	0,60375	10,1998	0,36373
9000	5,6218	4,384	1,28234	7,1453	0,78678	8,217	0,53352
10000	7,5116	5,682	1,32199	8,4684	0,88701	10,0951	0,56284

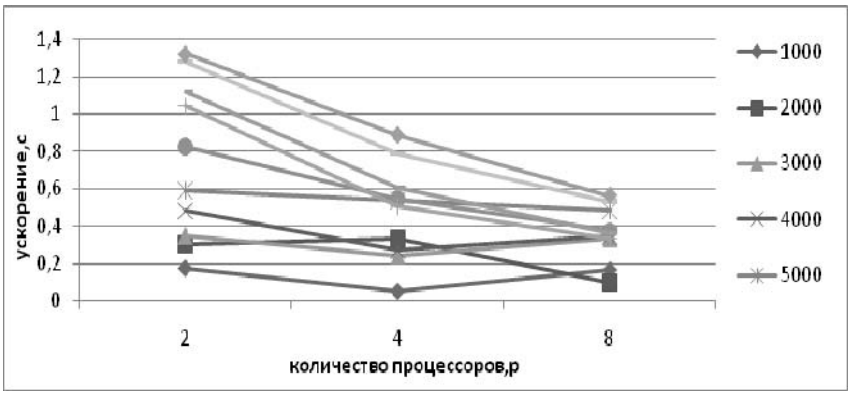

Рис. 6. Графики зависимости ускорения параллельного алгоритма Борувки от числа используемых процессоров при различном количестве вершин в модели

Результаты сравнения экспериментальной T^* и теоретической T оценки времени работы параллельного алгоритма Борувки в зависимости от количество вершин в модели при использовании 2,4,8 процессоров представлены в табл. 4, а при использовании 2 процессоров – на рис. 7.

Теоретическая оценка, проводилась по формуле:

$$T=n^2[n/p]*\tau+ n[log_2p](r+vn/q).$$

Таблица 4

Сравнение экспериментального и теоретического времени работы алгоритма Борувки

Количество вершин, n	Параллельный алгоритм Борувки					
	2 процессора,р		4 процессора,р		8 процессоров,р	
	T_2, с	T^*_2, с	T_4, с	T^*_4, с	T_8, с	T^*_8, с
1000	0,3754	0,2456	0,804	0,832	1,1891	1,4635
2000	0,8121	0,6821	1,5928	1,6421	2,3872	2,0499
3000	1,2335	1,4	2,4151	2,013	3,7216	3,9992
4000	1,6541	1,8123	3,1763	3,1797	4,699	5,1518
5000	1,973	2,423	3,9958	2,653	6,0249	4,989
6000	2,575	2,658	4,8518	4,001	7,124	7,0784
7000	3,0375	2,9212	5,7231	6,013	8,38	8,8255
8000	3,4875	3,71	6,4516	6,8731	9,652	10,1998
9000	4,33	4,384	7,3882	7,1453	10,89	8,217
10000	4,4791	5,682	8,2241	8,4684	12,1432	10,0951

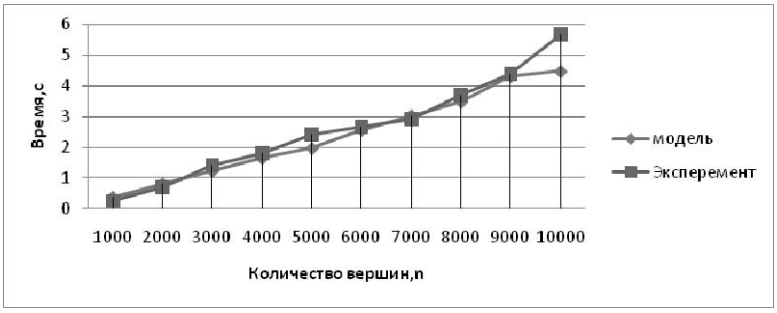

Рис. 7. Графики экспериментально установленного времени работы параллельного алгоритма Борувки и теоретической оценки в зависимости от количества вершин в модели при использовании 2 процессоров

Как видно из таблицы (см. табл. 4), разработанный метод на основе Борувки несколько лучше. Максимальное ускорение параллельного алгоритма Борувки составляет 1,32, а Прима – 1,29 и также в большинстве других случаев метод Боровки оказывался быстрее.

Выводы. Сравнительная характеристика экспериментальных и теоретических оценок параллельных алгоритмов нахождения минимального остовного дерева показала достаточную эффективность разработанного параллельного алгоритма на основе метода Борувки для нахождения минимального остовного дерева. Данный алгоритм может использован для разработки параллельной программы для кластера локальной сети, либо многопроцессорной вычислительной системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Интернет-ресурс http://www.winhpc.ru/?id=138(дата обращения 12.12.2010).
2. Интернет-ресурс http://rain.ifmo.ru/cat/view.php/theory/graph-spanning-trees/mst-2005(дата обращения 10.12.2010).
3. *Аль-хулайди А.А.* Разработка параллельного алгоритма для нахождения минимального остовного дерева // Шестнадцатая международная открытая научная конференция "Современные проблемы информатизации" издательство.– Воронеж: ВГТУ, 2011. – № 3. – С. 1-14.

Статью рекомендовал к опубликованию д.т.н., профессор Н.А. Целигоров.

Аль-Хулайди Абдулмаджид Ахмед
Донской государственный технический университет.
E–mail: abdulmajed_83@mail.ru.
344010, г. Ростов-на-Дону, ул. Мечникова, 79 а.
Тел.: +79896211028.
Профессор.

Чернышев Юрий Олегович
Донской государственный технический университет.
E–mail:pmivt@rgashm.ru.
344023, г. Ростов-на-Дону, пл. Страна Советов, 1.
Тел.: +79185991645.
Профессор.

Al-Khulaidi Abdulmajed Ahmed
Don State Technical University.
E–mail: abdulmajed_83@mail.ru.
79 a, Metshenkova Street, Rostov-on-Don, 344010, Russia.
Phone: +79896211028.
Professor.

Chernyshev Jury Olegovich
Don State Technical University.
E–mail:pmivt@rgashm.ru.
1, Strana Sovetov Street, Rostov-on-Don, 344023, Russia.
Phone: +79185991645.
Professor.

УДК 658.512.2.011.5

В.В. Лисяк, Н.К. Лисяк

МНОГОУРОВНЕВЫЕ МОДЕЛИ В АНАЛИЗЕ САПР*

Рассмотрена одна из задач системного уровня проектирования, которая позволяет изучить систему, где клиенты, обращающиеся в случайные моменты времени за услугами и требующие различного времени обслуживания, могут выстраиваться в очереди. Рассмотренные модели дают вероятностные распределения длины очереди, моментов поступления обращений и времен ожидания обслуживания. Эти параметры важны в системах, где потери из-за перегрузки могут быть скомпенсированы лучшей организацией обслуживания.

Ресурс; заявка; многоресурсная система; стохастическая сеть; обслуживающий аппарат; дисциплина очереди; многоуровневые системы; вложенный процесс.

V.V. Lisyak, N.K. Lisyak

MULTILEVEL MODELS FOR CAD SYSTEM ANALYSIS

There is considered a task of system level of design, which allows studying the system, where the clients request for services at random moments, and require different service time, and can line up in queue. Considered models provide the probability distribution of queue length, moments of requests' receiving, and times of waiting for service. These parameters are important in the systems, where losses caused by overloading can be compensated with better organization of service.

Resource; demand; multiresource system; stochastic network; service unit; queue discipline; multilevel systems; embedded process.

Введение. Многообразие ресурсов САПР и сложность их взаимодействия ставят задачу анализа моделей стохастической, сетевой структуры [1–4,8]. При этом использование одной задачей нескольких разнородных ресурсов приводит к одноуровневому или многоуровневому представлению взаимодействия ресурсов. В одноуровневом представлении такую задачу упрощают за счёт ввода в модель логических условий и блокировок, либо выделяют какой-либо один из совместно используемых ресурсов, а другие ресурсы либо не учитываются, либо учитываются введением некоторых ограничений. Эти упрощения можно обойти при многоуровневом представлении взаимодействия ресурсов, в котором одновременное занятие заявкой нескольких ресурсов отображается с помощью механизма вложенных процессов [4]. В таком подходе в моделях отдельных уровней отсутствуют сложные логические условия и блокировки, что способствует применению аналитических методов исследования. Построение и описание формализованной схемы осуществляется по уровням. Если параметры времени обработки заявки каким-то ресурсом не определены, то для этого ресурса строится более детальное описание процесса обработки на следующем уровне. Формализованная схема построена, если все временные параметры обработки заявок ресурсами определены.

* Работа выполнена при частичной поддержке РФФИ (проект № 11-01-00975).