Ответы на замечания по статье
«ПРИМЕНЕНИЕ МЕТОДА ПРОЕКТИРОВАНИЯ Q-ЭФФЕКТИВНЫХ ПРОГРАММ ДЛЯ АЛГОРИТМА ДЕЙКСТРЫ»

Авторы выражают благодарность рецензентам за ценные замечания, способствующие улучшению текста статьи.

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	Рецензент А	
	Содержание	
1.	Замечание 3 касательно рудиментарности раздела 3 не видится исправленным в полной мере. Раздел 3 по-прежнему выглядит рудиментарно, а добавленное авторами описание средств, используемых при реализации Q-эффективных программ, по сути, относится к имплементации метода проектирования программ, но не к самому методу.	Выполнено. Раздел 3, содержавший ранее описание метода проектирования Q-эффективных программ, удален. В новой редакции описание метода содержится в разделе 2, а описание средств разработки для его использования – в разделе 4.
2.	Исправление замечания 7 касательно подбора графов для проведения экспериментов и ответ авторов на него не видятся удовлетворительными.	Ответы даны далее по подпунктам.
2.1.	Граф(ы) для экспериментов не обязательно подбирать, ссылаясь только на те работы, в которых был реализован алгоритм Дейкстры с применением представления графа в виде матрицы смежности (хотя это и желательно). Подойдет любой релевантный задаче граф с фиксированной и описанной структурой, чтобы обеспечить репродуцируемость экспериментов.	См. ответ на замечание 3.
2.2.	Некорректен отказ от использования в экспериментах референсных графов в зависимости от способа их представления в соответствующей статье: представление данных - это "внутреннее дело" алгоритма, и это представление выбирается для достижения алгоритмом наибольшей эффективности.	См. ответ на замечание 3.
2.3.	Кроме того, в ответе на замечание авторы указывают, что в экспериментах проводили запуски "для разных графов с одинаковым числом вершин и в качестве результата" брали среднее арифметическое времени выполнения для исследования эффективности своего решения "в среднем случае". Поскольку указанные выше графы получаются случайным образом, то итоговое среднее значение эффективности поиска	На основании анализа полученных рядов измерений времени выполнения для определенных сочетаний конфигурации и размерности задачи можно сделать вывод, что значение времени выполнения подчиняется закону нормального распределения (точнее: нет оснований отвергать гипотезу о подчинении ряда этому закону распределения по критерию Пирсона). Это также справедливо и для ускорения, и для эффективности распараллеливания, т.к. они являются производными величинами от времени выполнения.

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	кратчайшего пути в них видится лишенным теоретической и практической ценности.	В данных рядах выборочная средняя (среднее арифметическое) почти равна выборочной медиане, а мода отсутствует, т.к. значения в рядах не повторяются. Тогда среднее арифметическое можно считать главной характеристикой, по которой можно определить центр распределения (наиболее вероятное значение). Таким образом, использование среднего арифметического однозначно имеет под собой теоретическое обоснование и практическую ценность.
3.	Исправление замечания 8 касательно необходимости сравнения своего подхода с аналогами и ответ авторов на него не видятся удовлетворительными.	Алгоритм Дейкстры предназначен для решения алгоритмической проблемы, заключающейся в вычислении кратчайших расстояний от выделенной вершины графа до всех остальных вершин. Если для решения этой алгоритмической проблемы использовать два разных способа представления входных данных, то мы имеем два разных алгоритма, так как из-за разных множеств их входных данных, являющихся подмножествами множества B, они имеют разные Q-термы и Q-детерминанты (см. раздел 2). При этом ресурсы параллелизма алгоритмов могут отличаться. Следовательно, могут отличаться и динамические характеристики реализующих их Q-эффективных программ. Наша статья посвящена разработке и исследованию Q-эффективных программ для алгоритма, реализующего алгоритм Дейкстры, который для представления входных данных использует матрицу смежности. При этом мы не ставим задачу сравнивать динамические характеристики разработанных Q-эффективных программ с динамическими характеристиками программ, которые реализуют алгоритмы, использующие другие представления входных данных. Решение этой задачи может быть результатом исследования для другой статьи. Динамические характеристики программ зависят от реализуемого алгоритма, точнее от его ресурса параллелизма, и от вычислительной инфраструктуры программы, т.е. от условий ее разработки и выполнения. Если зафиксирован алгоритм и вычислительная инфраструктура программы, то Q-эффективная программа имеет лучшие динамические характеристики среди всех программ, реализующих данный алгоритм и использующих данную вычислительную инфраструктуру. Так как вычислительных инфраструктур существует потенциально бесконечное множество, то для алгоритма не существует Q-эффективной программы с лучшими динамическими характеристиками. Если разработчик программы хочет улучшить ее динамические характеристики при той же вычислительной инфраструктуре, то он должен заменить алгоритм на алгоритм

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
		с меньшей высотой, используя для этого, например, Q-систему [23, 24]. Более подробно с данными исследованиями можно познакомиться с помощью [4] и работы Valentina N. Aleeva, Rifkhat Zh. Aleev. Parallelism Resource of Numerical Algorithms. Version 1. Arxiv.org [Электронный документ] (https://arxiv.org/pdf/2207.11915.pdf) 73 с. Теперь, что касается сравнения с аналогами статьи. Как и вычислительных инфраструктур, аналогов статей может существовать потенциально бесконечное множество. Если при сравнении динамических характеристик программ аналога и нашей статьи окажется, что аналог имеет динамические характеристики лучше, а алгоритм используется тот же, что в нашей статье, то это значит, что программы аналога имеют другую вычислительную инфраструктуру. Найти аналог статьи, в которой используется тот же алгоритм и та же вычислительная инфраструктура, что в нашей статье, практически невозможно, так как, если даже к полному тексту аналога статьи есть доступ, как правило, он не содержит описание всех характеристик вычислительной инфраструктуры. Сравнение с аналогами в каких-то случаях имеет значение, но не в нашем. Решение проблемы разработки для конкретного алгоритма и имеющейся в распоряжении разработчика вычислительной инфраструктуры самой эффективной программы имеет большое значение. Наша статья заключается в том, что мы показываем впервые на примере алгоритма на графах, как разработать для него самую эффективную программу для часто используемой вычислительной инфраструктуры. В аналогах статьи этого нет. Ссылки даны на список литературы, приведенный в статье.
3.1.	Выбор представления данных в алгоритме аналоге не может являться препятствием для сравнения (см. п. 2 выше). Кроме того, для любых параллельных алгоритмов-соперников сравниваемые показатели, очевидно, должны включать быстродействие и ускорение, исследованные на одних и тех же входных данных и аппаратных платформах (в последнем случае при невозможности – на аппаратных платформах со сходной пиковой производительностью).	См. ответ на замечание 3.
3.2.	В ответе на замечание авторы ссылаются лишь на два аналога, не опублико-	Выполнено. Ссылки на некоторые аналоги статьи приведены в разделе 1.

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	ванные в рецензируемых научных изданиях (по сути - слайды неких презентаций), что нельзя признать корректным аргументом. Следует провести более тщательный поиск научных публикаций и включить их в обзорный раздел. Например, рецензенту в библиографическом каталоге DBLP с помощью запроса https://dblp.org/search?q=dijkstra%20parallel удалось найти следующие достойные как минимум упоминания публикации: 1) Maria Fazio, Alina Buzachis, Antonino Galletta, Antonio Celesti, Jiafu Wan, Antonella Longo, Massimo Villari: A Map-Reduce Approach for the Dijkstra Algorithm in SDN Over Osmotic Computing Systems. Int. J. Parallel Program. 49(3): 347-375 (2021). https://doi.org/10.1007/s10766-021-00693-3. 2) Weidong Zhang, Lei Zhang, Yifeng Chen: Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor - How to Accelerate Irregular Memory Access Algorithm. ICA3PP (1) 2018: 337-357. https://doi.org/10.1007/978-3-030-05051-1_24. 3) Nadira Jasika, Naida Alispahic, Arslanagic Elma, Kurtovic Ilvana, Lagumdzija Elma, Novica Nosovic: Dijkstra's shortest path algorithm serial and parallel execution performance analysis. MIPRO 2012: 1811-1815. https://ieeexplore.ieee.org/document/6240942/.	
4.	Исправление замечания 9 касательно определения исследуемых в статье показателей ускорения и эффективности и ответ авторов на него не видятся удовлетворительными.	Ответы приведены в следующих подпунктах.
4.1.	Авторы по-прежнему неверно либо понимают, либо отражают на графиках ускорение параллельной программы, но и первое, и второе запутывает читателей и рецензентов, и потому неприемлемо. Измерение ускорения предполагает многократный запуск параллельной программы *на одном и том же наборе данных*, но на различном (увеличивающемся) количестве вычислительных	На рис.1 программы для общей памяти выполняются на одном узле на двух процессорах. Поэтому ось абсцисс на рис.1, обозначающая количество процессоров/узлов, отсутствует. Существует два основных определения ускорения: по Амдалу и по Густавсону. По Амдалу ускорение определяется на одной и той же размерности задачи при разном количестве процессоров и в англоязычных статьях оно называется “fixed-size speedup”. Ускорение по Густавсону

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	устройств (ядер процессора, узлов кластера и проч. - в зависимости от аппаратной платформы). Ускорение программы при переходе от одного к N выч. устройствам вычисляется как отношение T_N к T_1, где T_N – время выполнения программы на N устройствах, а для времени T_1 программу, запускаемую на одном устройстве, нужно отдельно оговорить и обосновать выбор: это может быть последовательная программа, реализующая наиболее быстрый последовательный алгоритм или программа, реализующая оригинальный последовательный алгоритм (без директив компилятора для автоматического распараллеливания) или оригинальная параллельная программа. График ускорения предполагает в оси абсцисс количество вычислительных узлов. Если авторы желают исследовать некую иную характеристику параллельной программы, то не следует ее называть ускорением, а подыскать другой термин. Ответ авторов, в котором утверждается, что "приводятся результаты для графов с одинаковым количеством вершин", и три графика в статье на рис. 1, где по оси абсцисс показано увеличивающееся количество вершин, окончательно все запутывает. Можно предположить, что авторы хотят показать в одном графике зависимость ускорения от размера графа, но неясно, какова практическая польза такого знания (если это действительно было целью экспериментов, то следует дать соответствующее обоснование), и совершенно некорректно называть этот показатель ускорением. Заметим также, если ускорение программы (в классическом смысле этого термина) при запуске на 24 нитях составляет максимум 1.8 (как показано на рис. 1б), то этот факт существенно диссонирует с тем, что программа называется Q-эффективной.	определяется при соразмерном увеличении размерности задачи и количества процессоров за постоянный промежуток времени (“fixed-time speedup”) и относится к такой предлагаемой Вами характеристике, как расширяемость.(Speedup for Multi-Level Parallel Computing (nus.edu.sg)) Существуют более конкретные определения ускорения, которые могут, например, содержать требование сравнения времени выполнения на одной и той же задаче. Но за их основу, как правило, берется одно из вышеприведенных определений. Этот момент был учтен в глоссарии в открытой энциклопедии свойств алгоритмов, где ускорение определяется абстрактно. (Глоссарий — Алговики (algowiki-project.org)) Таким образом, используемое в статье определение ускорения является корректным, как и его использование. Добавлено определение последовательной программы. Динамические характеристики Q-эффективной программы, в т.ч ускорение, зависят от ресурса параллелизма алгоритма. Другими словами, мы не можем обеспечить ускорение больше, чем допускает ресурс параллелизма алгоритма. См. также ответ на замечание 3. Изменение порядка осей возможно, но влечет собой заслонение большой части данных и потерю информативности. Изменение угла перспективы также не помогает. Примечание: Вами дано определение величины, обратной к ускорению, а не самого ускорения. Ведь предполагается, что на нескольких устройствах время выполнения меньше.
4.2.	Дополнительно можно напомнить авторам о рекомендации рецензента в первом раунде экспертизы статьи исследовать другую стандартную характеристику параллельной программы - рас-	Масштабируемость бывает трех видов: масштабируемость вширь (wide scaling), сильная (strong scaling) и слабая масштабируемость (weak scaling) (Глоссарий — Алговики (algowiki-project.org)). Таким образом, упомянутая Вами характеристика «расширяемость» по-другому называется слабой

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	ширяемость, когда предполагается соразмерное увеличение набора данных и количества выч. устройств.	масштабируемостью и не является единственной подобной характеристикой. В графике в) на рис. 2 действительно были неявно отражены такие характеристики, как сильная масштабируемость (если выбрать одно значение на оси «Количество вершин» и идти вдоль оси «Конфигурация») и масштабируемость вширь (если идти вдоль по оси «Количество вершин», выбрав одну из конфигураций). Упомянутые характеристики являются производными от эффективности распараллеливания и в контексте данного исследования не могут быть определены через другие величины. В статье термины данных характеристик не упоминаются, т.к. в описании результатов эксперимента они описываются в части, характеризующей эффективность распараллеливания. Исследовать слабую масштабируемость не представляется возможным в полной мере, т.к. для полного исследования требуется 240 вычислительных узлов, а квота пользователя суммарно составляет не более 70 узлов, работающих одновременно на всех задачах. Обоснуем, почему именно 240 узлов: эксперименты проводились на графах с числом вершин от 2 тыс. до 40 тыс. Размеры отличаются в 20 раз, а в экспериментах использовались максимум 12 узлов. Тогда для полного изучения слабой масштабируемости необходимо $12 * 20 = 240$ узлов.
4.3.	Также нужно отметить, что при определении ускорения и эффективности авторы выбрали весьма неудачную статью [https://doi.org/0.20537/2076-7633-2010-2-3-231-272] для ссылки, автор которой обозначает свой труд как "некоторые учебные материалы" и при этом ссылается на существенно более ранний учебник [Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб: БХВ-Петербург, 2002], который является в данном случае предпочтительным первоисточником для ссылки.	Выполнено.
5.	Измененные графики на рис. 1 и 2, к сожалению, не добавили ясности. Непонятно, для чего отдельно исследуется поведение параллельной программы на общей памяти, когда на процессоре запускается меньшее количество нитей, чем это можно сделать на текущей аппаратной платформе (6, 12, 18 нитей вместо 24): эффективность параллельной программы, очевидно, предполагает наиболее рациональное использование	В экспериментах задействуются все процессоры вычислительного узла, и это указано в статье. Таким образом, используются все доступные аппаратные ресурсы. Нить (как и процесс) являются абстракцией **конкретной** операционной системы и не относится к аппаратной платформе вычислительной системы. Количество нитей в экспериментах варьируется по следующей причине: нити по умолчанию не привязываются к конкретному ядру процессора и «мигри-

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	всех доступных ей аппаратных ресурсов. Еще более запутанно в указанном смысле исследование эффективности параллельной программы на распределенной памяти: непонятно, для чего отдельно исследовать поведение программы на маломощной части текущей аппаратной платформы (например, 3 узла и 6 нитей на каждом). Переработка результатов экспериментов, очевидно, потребует изменения текста в части обсуждения результатов. Возможно, у авторов получится представить результаты экспериментов на большем количестве узлов суперкомпьютера Торнадо.	руют» между ядрами, что часто порождает накладные расходы при планировании нитей **конкретной** операционной системой, установить влияние которых на ускорение программы в зависимости от числа нитей без проведения данного эксперимента не представляется возможным. Только таким образом мы смогли выявить закономерности, отраженные в п.3-5 в описании результатов эксперимента. Об отсутствии необходимости проведения экспериментов на большем числе узлов уже было сказано в ответе в предыдущей редакции рецензии. Напомним, что это обосновано получением в данном эксперименте крайне низких значений эффективности распараллеливания при отсутствии значительного роста ускорения.
6.	Авторы в ответе на замечание 13 о стилистических погрешностях пишут, что вычитали текст, однако прямо указанное в замечании неудачное предложение, содержащее две подряд конструкции "то есть" осталось без изменений.	Выполнено.

№ п/п	Замечание рецензента	Ответ авторов (отметка о выполнении и/или комментарий)
	Рецензент B	
	Содержание	
	1.Стр.2	
	Опечатка: "Проблемы эффективной реализации алгоритмов на ПВС посвящено много научных исследований."	Выполнено.
	2.	
	Раздел 3 довольно пустой. Авторы добавили туда описание использованных технологий, но не вполне понятно, какое они имеют отношение к сущности метода.	Выполнено. Раздел 3, содержавший ранее описание метода, проектирования Q-эффективных программ удален. В новой редакции описание метода содержится в разделе 2, а описание средств разработки для его использования – в разделе 4.
	3. Стр. 7	
	Ускорение – отношение времени выполнения последовательной программы к времени выполнения параллельной программы." Не сказано, что за последовательная программа берётся для сравнения. Или это параллельная программа, выполняемая одним процессом? "Эффективность распараллеливания – отношение ускорения к числу используемых параллельной программой процессоров." Поскольку процессоры многоядерные, правильно говорить о числе процессов (нитей).	В статью добавлено описание используемой последовательной программы. Последовательная программа выполняет на одном вычислительном узле ту же задачу, которая сформулирована при проектировании Q-эффективной программы, и не использует программные средства распараллеливания или межпроцессное взаимодействие. Процесс и нить являются абстракцией **конкретной** операционной системы и не относятся к аппаратной части вычислительной системы. О какой связи между ядрами процессора и процессами идет речь, к сожалению, неясно. На каждом узле выполняется только один процесс, отвечающий за выполнение изучаемой программы (в текст статьи добавлено это замечание). Описание зависимостей от числа процессов будет некорректно, т.к. в данной работе используется взаимодействие процессов на разных узлах, а не на одном и том же узле. Построенная таким образом зависимость не будет коррелировать с уже существующей, если число процессов будет больше числа узлов. Изучаемые характеристики также не стоит определять через количество нитей по одной причине: нити по умолчанию не привязываются к конкретному ядру процессора и «мигрируют» между ядрами, что часто порождает накладные расходы при планировании нитей **конкретной** операционной системой. С другой стороны, целесообразность привязки нитей «вручную» в большей степени зависит от архитектуры вычислительной системы (в особенности процессора и планировщика операционной системы), а не от выполняемой задачи. Это является предметом отдельного исследования. Соответственно, будет некорректно определять ускорение/эффективность через нити, которые по сути своей могут иметь разную реализацию на разных си-

		стемах (достаточно вспомнить про технологию гиперпоточности на процессорах суперкомпьютера «Торнадо ЮУрГУ», которой на других вычислительных системах может и не быть вовсе). Таким образом, использование предложенных Вами параметров для определения изучаемых характеристик влечет несопоставимость результатов, полученных в разных системах. Наконец, вернемся к тому факту, что нить и процесс являются абстракциями ОС, которые в будущих системах могут быть заменены на другие (например, на сопрограммы (coroutines) или механизм намерений (intents)). В таком случае определение характеристик через данные абстракции может повлечь собой скорое устаревание данной статьи.
4. Стр. 7-8:		
	На рис. 1 что-то не сходится. Если ускорение колеблется около 1.7 даже при использовании 24 нитей (рис. 1б), то это не может соответствовать эффективности распараллеливания около 0.85 (рис. 1в). Эффективность распараллеливания должна получиться 1.7/24, то есть, около 0.07. Если же авторы делят 1.7 на 2 (то есть, на число именно процессоров), то это совершенно неверно, поскольку распараллеливание ведётся между нитями. В графиках на рис.2 сложнее разобраться, но скорее всего, авторы тоже некорректно считают эффективность распараллеливания.	В статье дано определение ускорения и эффективности. Там речь идет только о количестве процессоров. Нити или узлы там не упоминаются. Напомним, что в экспериментах используются все процессоры задействованного узла и их ядра. Тогда расчет изучаемых характеристик от количества процессоров на узле позволяет оценить степень использования вычислительных ресурсов. Абстрагируясь таким образом от архитектуры вычислительной системы, мы получаем значения характеристик, которые вполне будут подлежать сравнению с результатами работы той же программы на других системах в дальнейшем.
5. Стр. 10:		
	Результаты обсуждаются по-прежнему малосодержательно. Авторы не отвечают на встающие вопросы, например: - Почему для общей памяти даже при использовании большого числа нитей получается ускорение только в 1.3-1.8 раза? Неужели структура алгоритма такая, что большего невозможно получить? - Почему для распределённой памяти ускорение не превосходит 6 даже на 12 узлах? - Почему авторы используют в экспериментах не более 2 нитей на вычислительный узел? Хотя авторы пишут, что вычислительные узлы "с центральными процессорами Intel Xeon X5680 с частотой 3.33 GHz, каждый из которых имеет 6 ядер и поддерживает 12 нитей". То есть, на	Ответы по порядку соответствуют вопросам: - Ускорение программы для общей памяти получилось небольшим, т.к. переход по вершинам осуществляется последовательно. Параллельно здесь осуществляется только инициализация массивов и итерации по формуле (1). Ускорение для программы на распределенной памяти выше, т.к. в данной программе этапы алгоритма в распределенных массивах осуществляются параллельно. - Если воспользоваться законом Амдала, максимально достижимое значение ускорения 6 при работе 18 процессоров на 9 узлах соответствует доле параллельных вычислений около 85%, что однозначно указывает на высокую степень использования ресурса параллелизма алгоритма. Так как описанная Q-эффективная реализация использует весь ресурс параллелизма алгоритма, то полученное значение ускорения можно считать значением предельно достижимого ускорения в данной вычислительной системе. - На рисунке 2 указанное число нитей задействовано на каждом узле. Данный момент ранее упоминался

	каждом узле нитей можно было запускать намного больше. Такое ощущение, что тут продолжается путаница между терминами "узел", "процессор", "ядро", "нить" и т.д. Не разобравшись с этим, крайне сложно объяснить результаты.	только в заголовках к рисункам, но с учетом предыдущих замечаний они были убраны. Описание эксперимента дополнено указанием количества задействованных нитей.