Московский государственный технический университет
им. Н.Э. Баумана

Домашнее задание по курсу
«Программирование параллельных процессов»

Вариант №17

Задача коммивояжера

Работу выполнил
студент группы ИУ7-33М
Кадыров Р.Р.
Работу проверил
Ковтушенко А.П.

Москва, 2018г.

Теоретическая часть

1. Постановка задачи.

Разработать программу для решения задачи коммивояжера. Построить на заданном графе оптимальный путь, проходящий через все вершины по одному разу и приходящий в исходную вершину. Исходные данные: массив вершин, массив ребер с приписанными весами. Оптимальный путь имеет оптимальный суммарный вес включенных в него рёбер.

Обосновать проектное решение (выбор алгоритма). Обеспечить равномерную загрузку процессоров. Результат вывести в текстовый файл. Исследовать зависимость времени счета от размерности задачи и количества процессоров.

2. Задача коммивояжера.

Задача коммивояжера является одной из самых известных задач комбинаторной оптимизации. Эта задача состоит в поиске наилучшего (выгодного, дешевого, краткого и т.п.) маршрута, проходящего через всех указанные города хотя бы по одному разу (чаще всего - с точностью по одному разу) с возвращением в начальный город.

Данная задача относится к классу **NP-трудных задач** и решение переборными методами (вычисление длин всех возможных маршрутов и их сравнение) практически невозможно (для nn городов существует (n−1)! /2(n−1)! /2 маршрутов). Например, если взять 10 городов - найдем 181440 путей, а для 20 - уже 6,1·10166,1·1016.

На практике чаще всего изучают **метод ветвей и границ** (и его модификации, в том числе алгоритм Литтла), который позволяет на каждом шаге отбрасывать целую группу неоптимальных маршрутов.

3. Метод ветвей и границ

Задачи дискретной оптимизации имеют конечное множество допустимых решений, которые теоретически можно перебрать и выбрать наилучшее (дающее минимум или максимум целевой функции). Практически же зачастую это бывает неосуществимо даже для задач небольшой размерности.

В методах неявного перебора делается попытка так организовать перебор, используя свойства рассматриваемой задачи, чтобы отбросить часть допустимых решений. Наибольшее распространение среди схем неявного перебора получил метод ветвей и границ, в основе которого лежит идея последовательного разбиения множества допустимых решений. На каждом шаге метода элементы разбиения (подмножества) подвергаются анализу – содержит ли данное подмножество оптимальное решение или нет. Если рассматривается задача на минимум, то проверка осуществляется путем сравнения нижней оценки значения целевой функции на данном подмножестве с верхней оценкой функционала. В качестве оценки сверху используется значение целевой функции на некотором допустимом решении. Допустимое решение, дающее наименьшую верхнюю оценку, называют рекордом. Если оценка снизу целевой функции на данном подмножестве не меньше оценки сверху, то рассматриваемое подмножество не содержит решения лучше рекорда и может быть отброшено. Если значение целевой функции на очередном решении

меньше рекордного, то происходит смена рекорда. Будем говорить, что подмножество решений просмотрено, если установлено, что оно не содержит решения лучше рекорда.

Если просмотрены все элементы разбиения, алгоритм завершает работу, а текущий рекорд является оптимальным решением. В противном случае среди нерассмотренных элементов разбиения выбирается множество, являющееся в определенном смысле перспективным. Оно подвергается разбиению (ветвлению). Новые подмножества анализируются по описанной выше схеме. Процесс продолжается до тех пор, пока не будут просмотрены все элементы разбиения.

Практическая часть

4. Модели для параллельных вычислений

- Модель OpenMP – в рамках данной модели для решения задачи между потоками существует общая очередь, содержащая частично развернутые ветви. Различные потоки продолжают удалять их из очереди, оценивая, могут ли они привести к оптимальным решениям или нет, и если да, расширяя их в дополнительные подзадачи и добавляя их в очередь.
- Модель MPI - имеется главный узел, который поддерживает информацию обо всех других (ведомых) процессорах и отслеживает, какие из них простаивают. Ведомые процессоры обрабатывают подзадачи; и когда им нужно создать новые ведомые процессоры, они главный узел о выделении новых процессоров. Главный узел также периодически обновляет у ведомых узлов информацию о лучшем решении на текущий момент.
- Гибридная модель – данная модель сочетает в себе преимущества обеих вышеупомянутых стратегий. На каждом подчиненном узле параллельно выполняются различные потоки (на нескольких ядрах этого узла). Узел имеет общую очередь среди потоков, и потоки работают параллельно с помощью OpenMP. Как и в случае модели MPI, ведущий узел отслеживает простаивающие ведомые узлы и позволяет другим ведомым узлам создавать их по мере необходимости. Именно эта модель будет использоваться для реализации параллельного алгоритма решения задачи коммивояжера методом ветвей и границ.

5. Результаты исследования

На рисунке 1 представлен результат запуска программы для 8 вершин на 2, 3 и 4 узлах.

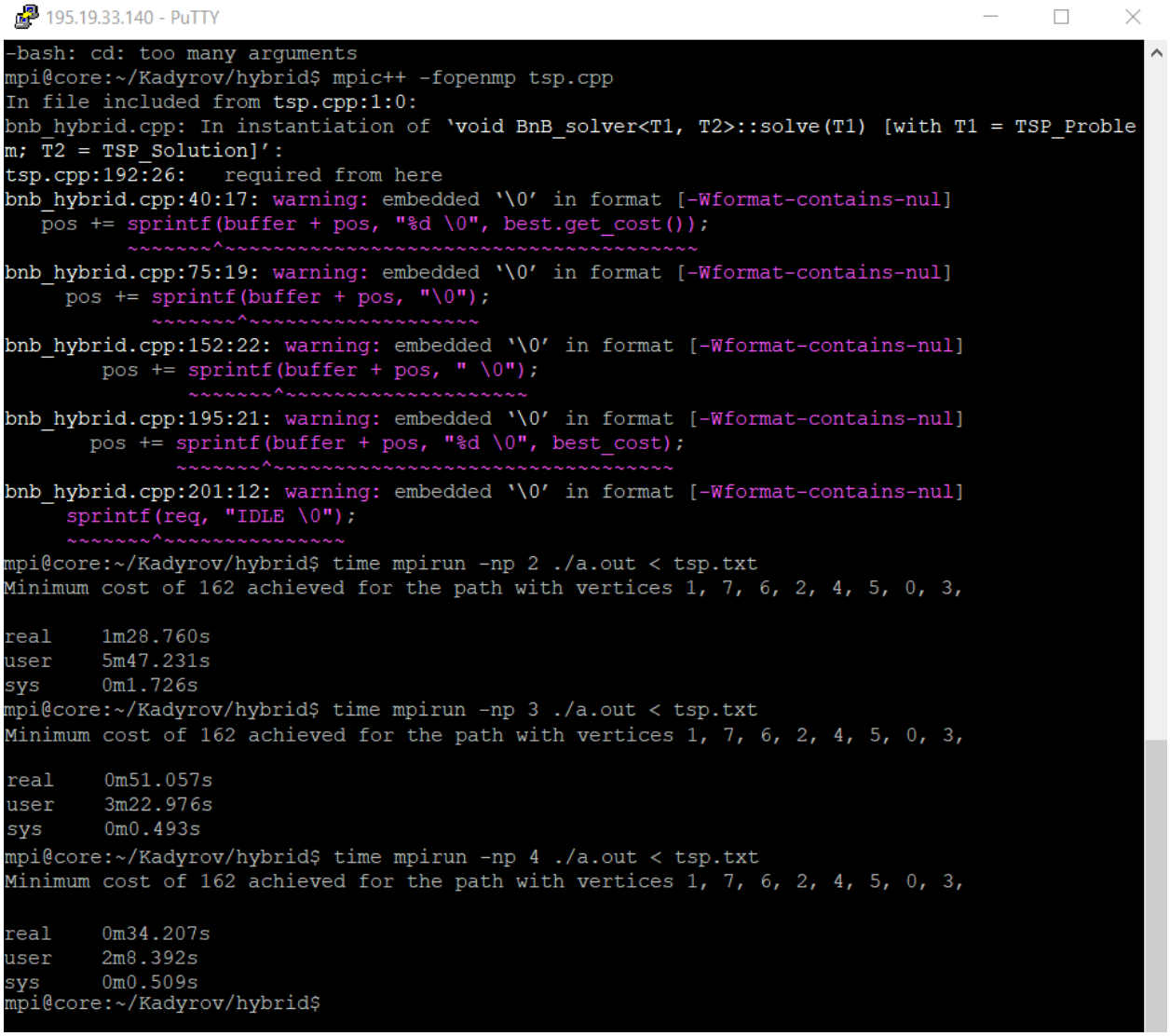

Рис. 1. Результат запуска программы для 8 вершин на 2, 3 и 4 узлах кластера.

Для проверки работы программы было подсчитано время в секундах, требующееся для решения задачи размером 8, 12 и 17 вершин на 1, 2, 3, 4 и 8 узлах, а также три метрики:

- отношение времени работы программы на одном узле ко времени работы на нескольких узлах;
- КПД одного процессора во время исполнения программы;
- отношение времени работы к числу узлов.

Результаты этих исследований представлены в таблицах 1 - 4 соответственно.

Число процессов *Число вершин*	**1**	**2**	**3**	**4**	**8**
8	103,14	87,760	51, 057	34,207	21, 241
12	452,98	329,026	232,821	184,243	121,901
17	934,32	707,439	594,211	432,358	253,129

Таблица 1. Зависимость времени работы программы от числа процессоров в секундах.

График 1. Зависимость времени работы программы от числа процессоров в секундах.

Число процессов *Число вершин*	**1**	**2**	**3**	**4**	**8**
8	1	1,1752	2,020095	3,01517	4,85703
12	1	1,3767	1,945623	2,45860	3,71597
17	1	1,3207	1,572373	2,16098	3,69108

Таблица 2. Отношение времени выполнения программы выполнения на одном узле к времени на нескольких узлах.

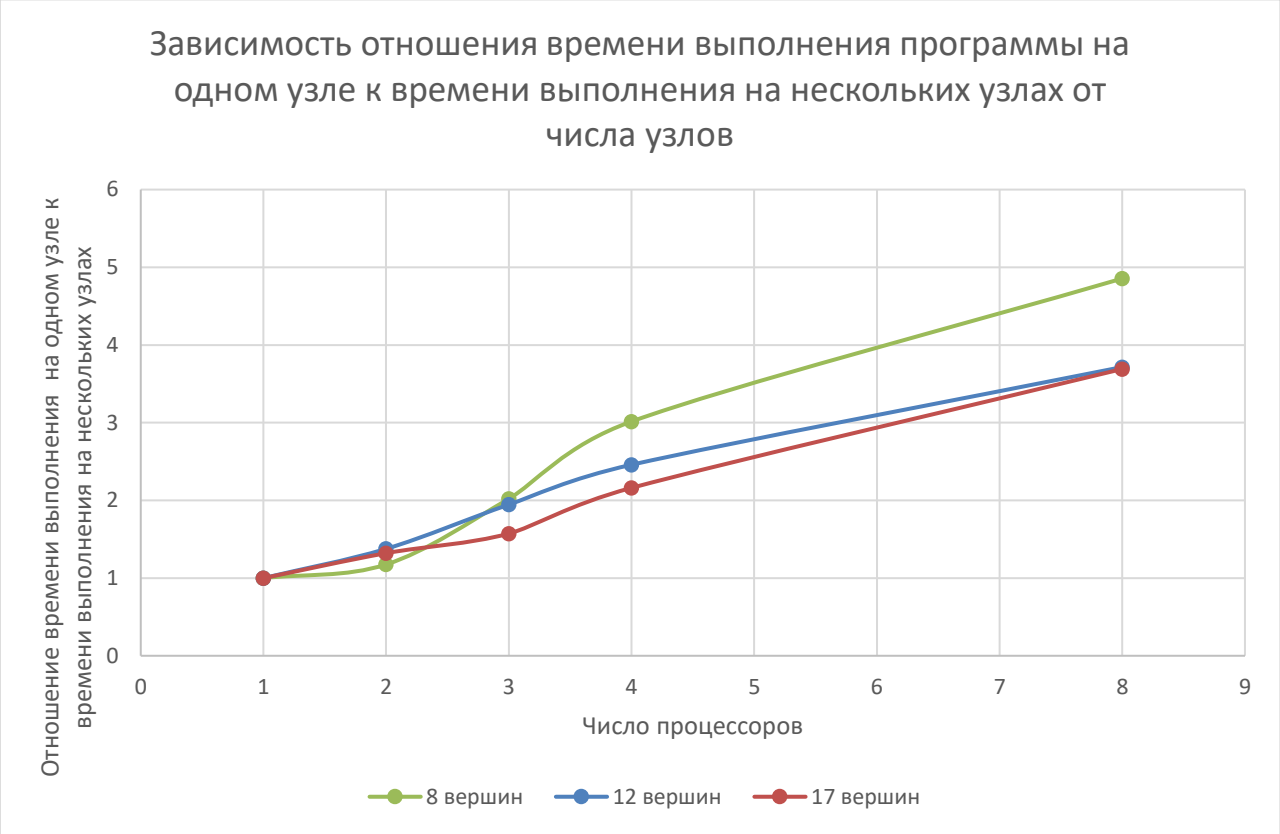

График 2. Зависимость отношения времени выполнения программы на одном узле к времени выполнения на нескольких узлах от числа узлов.

Число процессов *Число вершин*	**1**	**2**	**3**	**4**	**8**
8	100%	58,76%	67,34%	75,38%	60,69%
12	100%	68,84%	64,85%	61,47%	46,45%
17	100%	66,03%	52,41%	54,02%	46,14%

Таблица 3. Относительное КПД одного процессора.

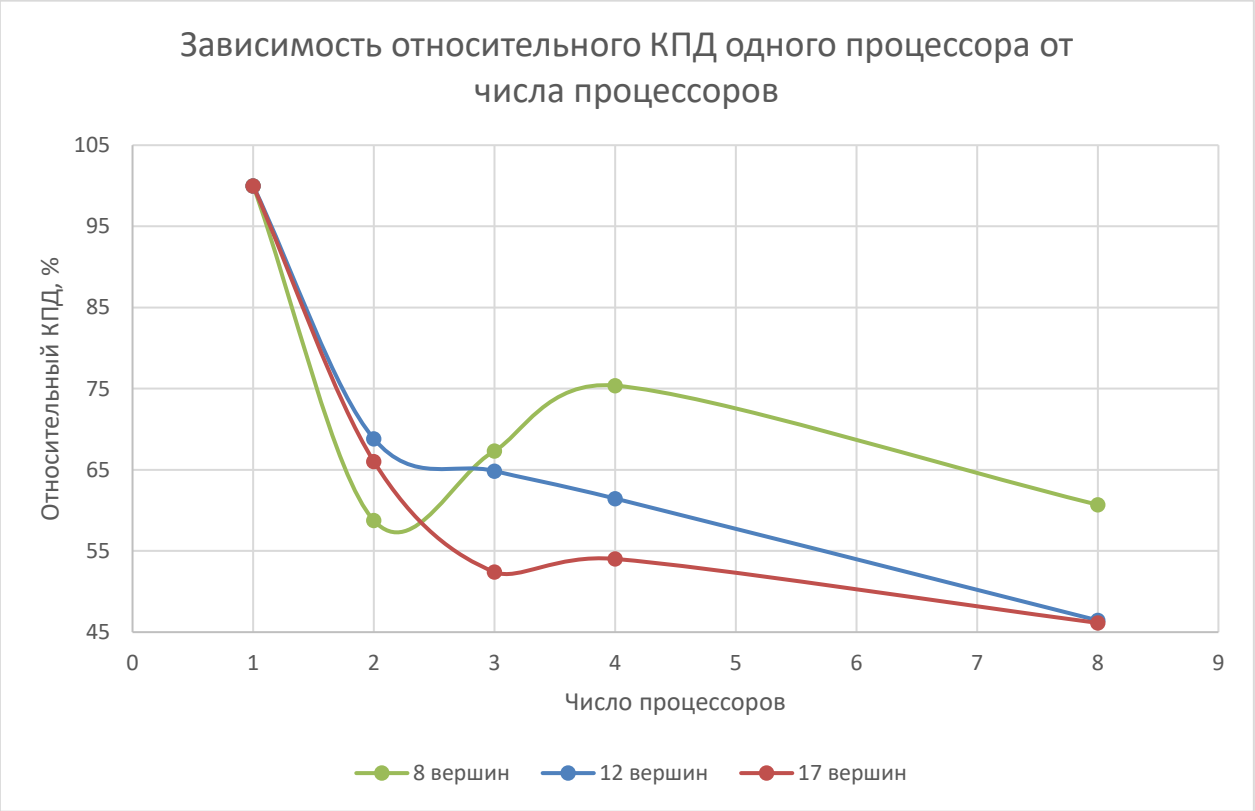

График 3. Зависимость относительного КПД одного процессора от числа используемых процессоров.

Число процессов *Число вершин*	**1**	**2**	**3**	**4**	**8**
8	113,14	43,88	17,019	8,55175	1,530125
12	452,98	164,513	77,607	46,06075	15,237625
17	934,32	353,7195	198,0703	108,0895	31,641125

Таблица 4. Отношение времени выполнения программы к числу используемых узлов.

Как было указано, используемая модель вычисления использует как MPI, так и OpenMP. Для ускорения работы была добавлена опция OpenMP #pragma parallel for. В ходе эксперимента использовалось 4 потока.

6. Выводы

В результате проведенной работы был реализован параллельный алгоритм метода ветвей и границ для решения задачи коммивояжера. Использованный алгоритм распределения столбцов позволил добиться равномерной загрузки всех процессоров и ускорения программы в ~ 4 раза. Было определено, что относительное КПД одного процессора при работе программы на нескольких узлах примерно равно 50%, а в некоторых случаях достигает 75%.