Санкт-Петербургский государственный университет

Кратчайший маршрут обхода отмеченных вершин графа

Выполнил:
студент 221 группы
Коротченок Остап Андреевич

Научный руководитель:
(доцент, кандидат физ-мат наук) Н. С. Григорьева

Отметка о зачете:

Санкт-Петербург 2021

Содержание

1 Введение

1.1 Постановка задачи

Задан ориентированный граф с n вершинами, на дугах графа задана длина дуги. Выделено множество помеченных вершин m и начальная вершина пути. Требуется обойти все отмеченные вершины и вернуться в начальную вершину кратчайшим маршрутом. Задача похожа на задачу о коммивояжере, но в этой постановке не запрещается пройденную вершину посещать вторично. Особенностью задачи является то, что если между двумя вершинами есть дуга, то обязательно есть и обратная, но длины этих дуг отличаются в несколько раз. Требуется разработать эвристические алгоритмы решения этой задачи.

1.2 Переформулировка задачи

Пусть V - множество вершин графа, M - множество отмеченных вершин, E - множество ребер графа. Очевидно, $|V| = n$, $|M| = m$. Сразу стоит заметить, что начальную вершину можно добавить ко множеству M, так как она тоже обязательна к посещению при обходе. Если заданный граф не является связным, то можно рассмотреть две ситуации.

1. $\exists a, b \in M$, что они лежат в разных компонентах связности. В таком случае у задачи нет решения.

2. $\forall a \in M$ принадлежат одной компоненте связности. В таком случае решение задачи сводится к ее решению в этой компоненте связности.

Теперь, не умаляя общности можно сказать, что решением задачи является поиск опимального пути в связном графе. Также требуется заметить, что если в связном графе $\forall a_1, a_2 \in V \mid \exists\ e_1 \in E$, соединяющее вершину a_1 с вершиной a_2, то $\exists\ e_2 \in E$, соединяющее вершину a_2 с вершиной a_1, то $\forall b_1, b_2 \in V\ \exists$ простой путь из вершины b_1 в вершину b_2.

1.3 Сведение задачи к поиску оптимальной перестановки

Пронумеруем вершины из M, т.е. $M = \{a_0, \dots, a_{m-1}\}$. Рассмотрим оптимальное решение. Его можно записать как $v_0 \to \cdots \to v_l$ и среди пройденных вершин можно выбрать по одной каждую вершину из M, т.е. оптимальное решение имеет вид $v_0 \to \cdots \to a_{i_0} \to \cdots \to a_{i_{m-1}} \to \cdots \to v_l$, где $\forall i_j, j = 1 \dots m-1 \quad a_{i_j} \in M$. Зафиксируем i_k и i_{k+1} и рассмотрим промежуток оптимального пути $a_{i_k} \to \cdots \to a_{i_{k+1}}$. Утверждается, что этот путь является кратчайшим между вершинами a_{i_k} и $a_{i_{k+1}}$. Действительно, если бы это было не так, его можно было бы заменить на кратчайший, тем самым уменьшив суммарный вес всех ребер в пути, но тогда мы приходим к противоречию, ведь выбранный путь является оптимальным. Тогда суммарный вес ребер оптимального решения численно равен $\sum\limits_{j=0}^{m-2} K_{a_{i_j},\ a_{i_{j+1}}} + K_{a_{i_{m-1}},\ a_{i_0}}$, где $K_{a_{i_j},\ a_{i_{j+1}}}$ - вес кратчайшего пути из a_{i_j} в $a_{i_{j+1}}$, т.е. величина оптимального пути определяется перестановкой $(i_0\ \dots\ i_{m-1})$. Далее будем говорить, что суммарный вес ребер, соответствующий перестановке, есть сумма весов кратчайших путей между пронумерованными вершинами из множества помеченных, порядок индексов которых определен перестановкой. Также введем понятие фитнес функции, соответствующей перестановке: $f(i_0, \dots, i_{k-1}) = \sum\limits_{j=0}^{k-2} K_{a_{i_j},\ a_{i_{j+1}}} + K_{a_{i_{m-1},\ a_{i_0}}}$. Очевидно, любое оптимальное решение имеет минимальную фитнес функцию среди возможных при $k = m$.

2 Алгоритмы для решения задачи

2.1 Метод ветвей и границ

2.1.1 Описание

Метод ветвей и границ является методом нахождения оптимального решения задач дискретной оптимизации. По своей сути он является развитием полного перебора, но отличается от последнего тем, что с помощью специфичных для каждой задачи эвристик отсеивает решения, не являющиеся оптимальными.

Такой алгоритм не имеет асмптотической сложности, но работает быстрее полного перебора.

2.1.2 Математическое описание

Определим путь с минимальнной фитнес функцией среди рассмотренных как лучший на данном этапе. Пусть рассматривается неполный путь $(i_0 \ \dots \ i_k), k < m - 1$. Тогда далее будут рассмотрены все пути $(i_0 \ \dots \ i_k \ i_{k+1})$, где i_{k+1} еще не была пройдена, которые удовлетворяют ряду условий.

1. Если вес изначальной перестановки плюс вес кратчайшего пути в новую вершину больше фитнес функции от лучшего пути, переход в эту вершину не рассматривается.

2. Если фитнес функция от полученной перестановки больше фитнес функции от лучшего пути, дальше она не рассматривается.

Если совершается переход от перестановки $(i_0 \ \dots \ i_k)$ к перестановке $(i_0 \ \dots \ i_k \ i_{k+1})$, то все вершины $a_l \in M$, содержащиеся в кратчайшем пути из a_{i_k} в $a_{i_{k+1}}$ помечаются как пройденные. Т.е. если кратчайший путь из a_{i_k} в $a_{i_{k+1}}$ имеет вид $a_{i_k} \to \cdots \to a_l \to \cdots \to a_{i_{k+1}}$, то вершина a_l считается пройденной, даже если $i_j \neq l \ \forall \ j \in \{0, \dots, k+1\}$.

2.1.3 Псевдокод

Algorithm 1 Branch and Bound

function Branch and Bound(path)

```
1: let dist be weight(path)
2: let l be the last element of current path
3: if path is finished and fitness(path) < fitness(best path) then
4:     best path = path
5: else
6:     for each v_i ∈ M | v was not visited do
7:         let current path be a copy of path
8:         if dist + weight(shortest path from l to v) < fitness(best path)  then
9:             append current path ← v
10:            if fitness(current path) < fitness(best path) then
11:                for each u ∈ M that is in shortest path from l to v do
12:                    set u as visited
13:                end for
14:                set v as visited
15:                call Branch and Bound(current path)
16:            end if
17:        end if
18:    end for
19: end if
```

Return best path

2.2 Жадный Алгоритм

2.2.1 Описание

Алгоритм заключается в принятии локально оптимальных решений, что в рамках данной задачи не приводит в общем случае к нахождению оптимального решения. Однако такой алгоритм работает чрезвычайно быстро в сравнении с другими предложенными в данной работе.

2.2.2 Математическое описание

Пусть рассматривается неполный путь $(i_0 \ \dots \ i_k), k < m - 1$. Тогда далее будут рассмотрены все пути $(i_0 \ \dots \ i_k \ i_{k+1})$, где i_{k+1} еще не была пройдена. Среди них выберем такое i_{k+1}, что кратчайший путь от i_{k+1} до него будет минимальным среди кратчайших путей от i_{k+1} до каждой из рассмотренных вершин, и перейдем к рассмотрению пути $(i_0 \ \dots \ i_k \ i_{k+1})$. Все вершины $a_l \in M$, содержащиеся в кратчайшем пути из a_{i_k} в $a_{i_{k+1}}$ помечаются как пройденные. Т.е. если кратчайший путь из a_{i_k} в $a_{i_{k+1}}$ имеет вид $a_{i_k} \to \dots \to a_l \to \dots \to a_{i_{k+1}}$, то вершина a_l считается пройденной, даже если $i_j \neq l \ \forall \ j \in \{0, \dots, k+1\}$.

2.2.3 Псевдокод

Algorithm 2 Greedy Algorithm

```
for each s ∈ M do
    set s as visited
    append path ← s
    while ∃ v ∈ M that is not visited do
        let l be the last element of path
        let min dist = ∞
        for each u ∈ M that is not visited do
            if weight(shortest path from l to u) < min dist then
                next = u
                min dist = weight(shortest path from l to u)
            end if
        end for
        for each a ∈ M that that is in shortest path from l to v do
            set a as visited
        end for
        set v as visited
        append path ← v
    end while
    if fitness(path) < fitness(best path) then
        best path = path
    end if
end for
```

Return best path

2.3 Алгоритм Муравьиной Системы

2.3.1 Описание

Алгоритм муравьиной системы основан на имитации естественного поведения муравьев в колонии, так как зачастую результат их деятельности приближается к оптимальному. В основе алгоритма лежит способ обмена информации между муравьями через изменение окружающей среды – муравьи оставляют на пройденном пути особый секрет (феромоны), который сохраняется там на протяжение какого-то времени. Другие муравьи могут определять концентрацию феромона на данном участке, чем она больше – тем более привлекательным является участок. В итоге по участкам с большей концентрацией феромона пройдет большее число муравьев. Важно отметить, что феромонная метка, оставленная любым муравьем, доступна любому муравью и что феромонная метка постепенно исчезает.

2.3.2 Математическое описание

Перед началом работы алгоритма требуется задать ряд констант:

1. Количество итераций $Iterations \in \mathbb{N}$
2. "Стадность"алгоритма $\alpha \in \mathbb{R}^+$
3. "Жадность"алгоритма $\beta \in \mathbb{R}^+$

4. "Перспективность"алгоритма $\gamma \in \mathbb{R}^+$
5. Изначальное количество феромона в участках системы $Pheromones \in \mathbb{R}^+$
6. Коэффициент испарения $Evaporation \in (0, 1)$
7. Количество муравьев $k \in \mathbb{N}$

Под участком системы будем подразумевать кратчайший путь между каждой парой выделенных вершин. Т.о. мы совершаем переход к полному графу $m \times m$, где ребро между любыми двумя вершинами по весу равно суммарному весу кратчайшего пути между выделенными вершинами в изначальном графе. Тут и далее в этом разделе будем ребром называть участок системы.
Инициализация системы. Уровень феромонов каждого ребра становится равен базовому, текущее значение феромонов на ребре будем обозначать $P_{i,j}$.
Итерация. Сначала обновляются параметры системы. К феромонам каждого ребра добавляются приобретенные за предыдущую итерацию феромоны $(Add_{i,j})$), а результат домножается на коэффициент испарения $P_{i,j} = Evaporation \cdot (P_{i,j} + Add_{i,j})$. Муравьи расставляются на начальные позиции. Каждый муравей независимо строит путь, возвращаясь в изначальную позицию, пройдя гамильтонов цикл по отмеченным вершинам. Путь строится последовательным выбором ребер и переходом в соответствующую вершину графа. Вероятность перехода по ребру r_k с весом d_k, перспективностью c_k и концентрацией феромонов P_k вычисляется по формуле:

$$Possibility_k = \frac{P_k^\alpha \cdot (\frac{1}{d_k})^\beta \cdot c_k^\gamma}{\sum_{k=1}^{l} (P_k^\alpha \cdot (\frac{1}{d_k})^\beta \cdot c_k^\gamma)} \tag{1}$$

Перспективность перехода от вершины i к вершине j вычисляется по формуле: $c_{i,j} = \sum_{v \in M} b(v)$, где $b(v) = 1$, если v есть в кратчайшем пути между вершианми i и j, и $b(v) = 0$, если отсутствует.

2.3.3 Псевдокод

Algorithm 3 Ant Colony Algorithm

```
let Ants be an array of n ants
let cnt = 0
while cnt < Iterations do
    Update Pheromones
    for each ant in Ants do
        Process(ant)
        if fitness(path(ant)) < fitness(best path) then
            best path = path(ant)
        end if
    end for
    cnt = cnt + 1
end while
Return best path
```

3 Применение и сравнение

3.1 Предварительная обработка входных данных

В качестве входных данных программа принимает матрицу смежности заданного графа и список помеченных вершин. В данной работе для этого генерировались связные графы с количесвтом ребер приблизительно в шесть раз меньшим, чем в полном. Весами ребер являлись случайные натуральные числа от 1 до 999 в общем случае. Также для метрического случая в качествt координат вершин выбирались случайные натуральные числа в том же диапазоне. В таком варианте весом ребра являлось расстояние между точками на плоскости в стандартной евклидовой метрике.
Перед запуском алгоритмов также на основе матрицы смежности создавалась матрица размера $m \times m$, где m - голичество выделенных вершин. В ней i-й строке j-м столбце хранился суммарный вес кратчайшего пути из вершины a_i в вершину a_j, где $a_i,\ a_j \in M$, а также список вершин $v \in M$, которые присутствуют в этом кратчайшем пути.
Константы для алгоритма муравьиной колонии можно подбирать для каждой матрицы смежности отдельно. В реализации есть страндартные значения констант, а также специальный алгоритм для их подбора. Он реализован как генетический, где в качесте хромосомы берется вектор констант, а в качестве фитнес функции - результат работы алгоритма муравьиной колонии с такими константами от одних и тех же входных данный и с малым количеством итераций. Ввиду чрезвычайной длительности работы такого алгоритма, его можно не запускать и выставить стандартные значения констант. При вычислении времени работы алгоритма муравьиной колонии время работы генетического алгоритма не учитывается.

n	m	Branch and Bound	Greedy Algorithm	Ant Colony Algorithm	
50	5	2801	2801	2801	Result
50	5	0.0013762	0.000143	0.340663	Time
50	10	2587	2688	2587	Result
50	10	2.46166	0.0004156	1.69296	Time
50	11	2915	3234	2921	Result
50	11	10.0237	0.0005013	2.02998	Time
50	12	3852	4638	3852	Result
50	12	93.8692	0.0008036	2.85406	Time
100	10	2231	2231	2231	Result
100	10	7.26716	0.0005294	2.18897	Time
100	15		3933	3683	Result
100	15		0.0012414	5.08227	Time
100	20		3578	3046	Result
100	20		0.0022181	9.3965	Time
100	40		6288	6131	Result
100	40		0.0131154	65.8446	Time

3.2 Сравнение результатов работы алгоритмов

4 Заключение

Метод ветвей и границ в данной реализации показал себя непрактичным при $m12$ из-за около-факториального времени работы. Это связано с недостатком эвристик, что приводит к отсеиванию недостаточного числа неоптимальных решений на ранних этапах. Если бы для любых $\forall a_1, a_2 \in M$ вес кратчайшего пути из a_1 в a_2 был бы равен весу кратчайшего пути из a_2 в a_1, то можно было бы использовать как оценку снизу минимальное остовное дерево на непройденных вершинах, но в общем случае ее использовать не представляется возможным.

Жадный алгоритм нельзя оценить в общем случае, но на основе полученных результатов можно сказать, что при малых m он часто находит оптимальные решения, а при больших m его результат не сильно превосходит результаты алгоритма муравьиной колонии. Также надо заметить, что время его работы чрезвычайно невелико.

Алгоритм муравьиной колонии стабильно дает решения ближе к оптимальным чем жадный, однако время его работы сильно увеличивается с ростом m. Его точность можно увеличить подбором констант для конкретных входных данных, а также увеличением количества итераций, но все это скажется на времени работы.

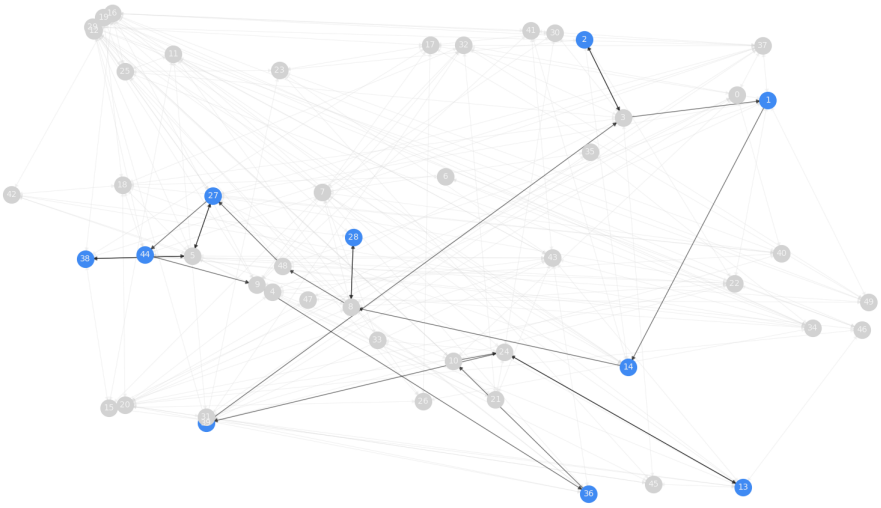
5 Список литературы

1. А.П. Карпенко 2017 «Современные алгоритмы поисковой оптимизации»: 147 – 193

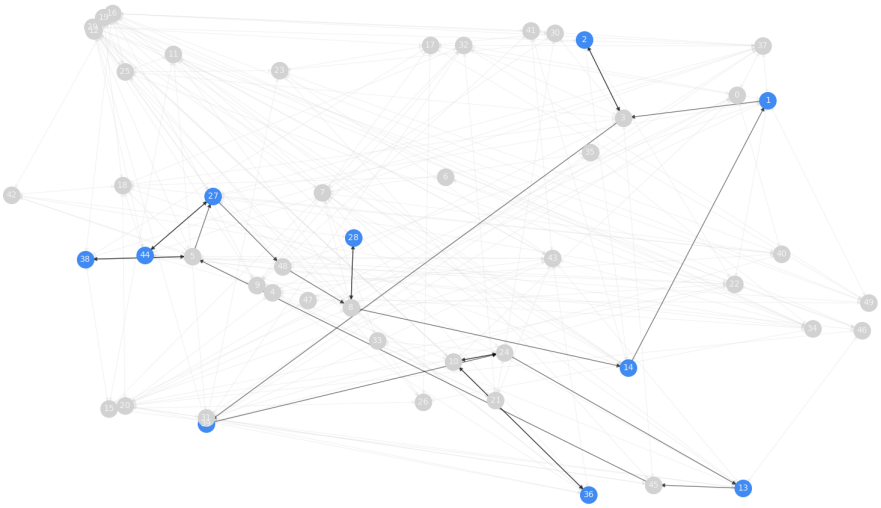
2. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. «Введение в алгоритмы»: 1111

6 Приложение

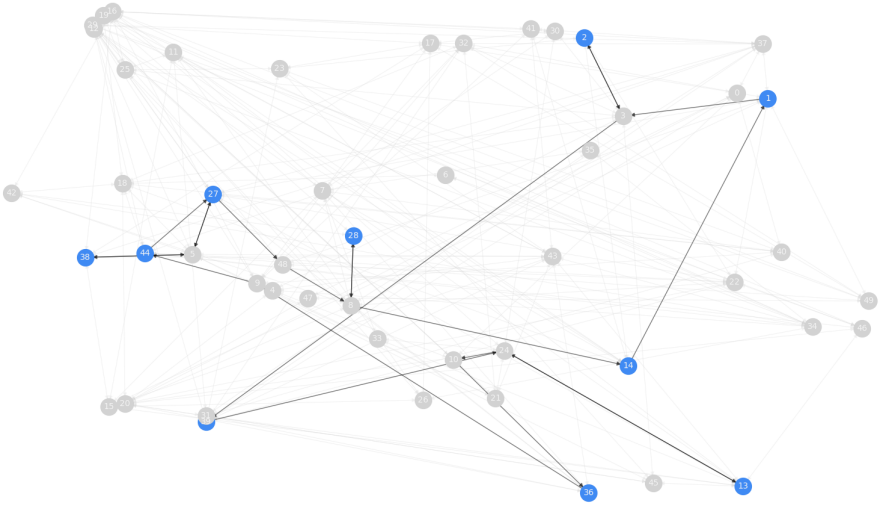
Некоторые результаты с метрическими графами в качестве входных данных. Визуализация выполнена с помощью Python и библиотек networkx и matplotlib.



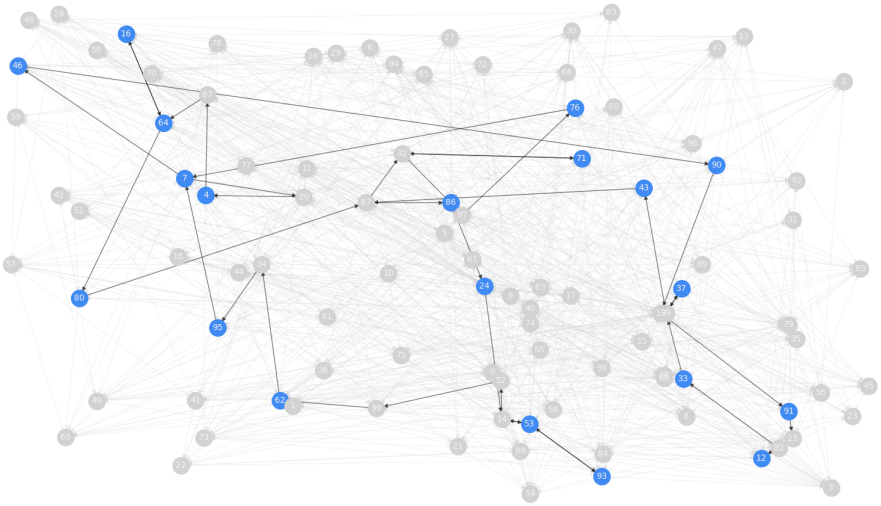
Branch and Bound: 5693



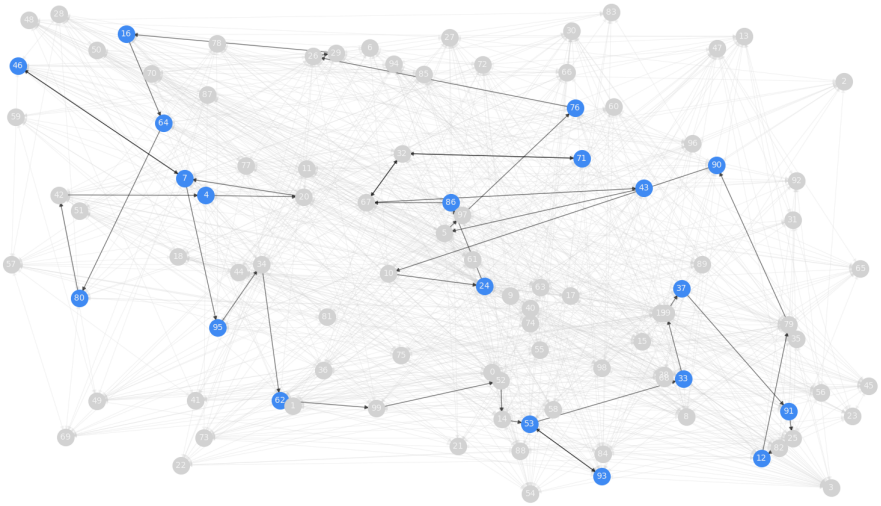
Greedy Result: 5789



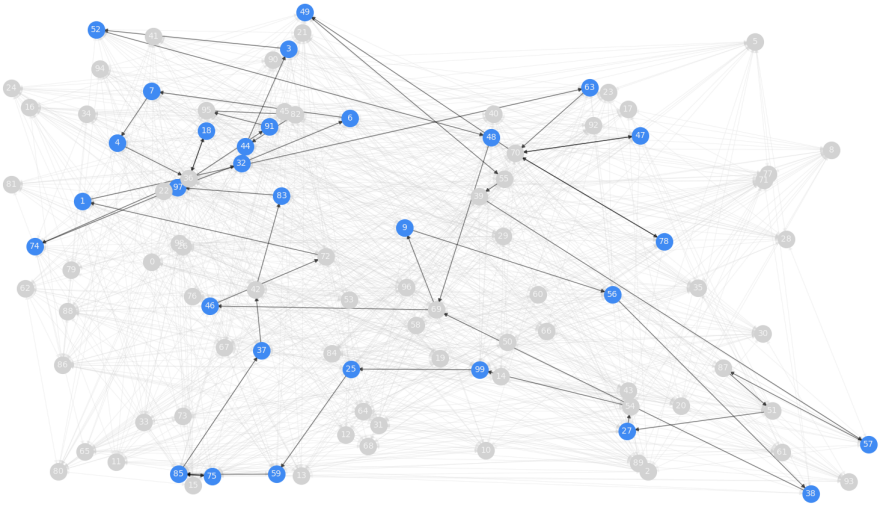
Ant Colony Result: 5693



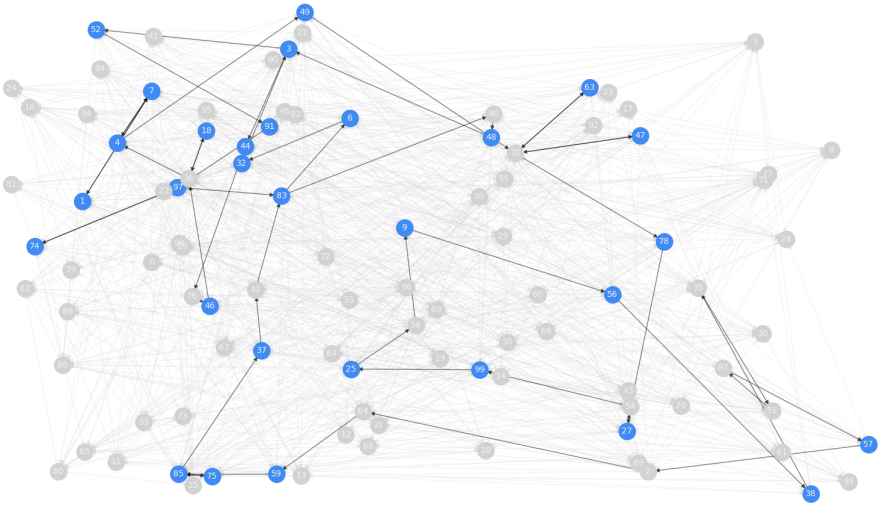
Greedy Result: 7697



Ant Colony Result: 7283



Greedy Result: 10419



Ant Colony Result: 10270