Серия «Математика»
2011. Т. 4, № 2. С. 114—123

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 519.854.2

Построение минимального остова с ограниченной пропускной способностью методом имитации отжига

А. В. Ипатов
Уральский государственный университет им. А. М.Горького

Аннотация. В статье рассматривается задача о минимальном остове с ограниченной пропускной способностью (CMST), относящаяся к классу NP-трудных. Разработан модифицированный метод имитации отжига, позволяющий получать лучшие решения CMST, чем классическая версия метода. Приводятся результаты вычислительного эксперимента.

Ключевые слова: минимальный остов, имитация отжига, метаэвристика, окрестность.

1. Введение

Пусть дан неориентированный граф $G = (V, E)$, где $V = \{v_0, v_1, \ldots, v_n\}$, $E = \{\{v_i, v_j\} \mid i \neq j\}$ и для каждого ребра из E определена его неотрицательная стоимость d_{ij}. Назовём вершину v_0 *корнем*, а все остальные вершины — *терминалами*. Будем считать, что для каждого терминала определён его *вес* q_i — неотрицательное действительное число.

Пусть T — остов графа G с корнем в v_0. Назовём *шлюзом* терминал, смежный с корнем в графе T, а *шлюзовым поддеревом* — максимальное корневое поддерево дерева T с корнем в шлюзе. Зафиксируем действительное число $Q \geqslant \max\limits_{v_i \in V \setminus v_0} q_i$, называемое *пропускной способностью шлюза*.

Задача о минимальном остове с ограниченной пропускной способностью (далее CMST = Capacitated minimum spanning tree) состоит в нахождении такого остова T графа G с корнем в v_0, что:

1) для любого шлюза T сумма весов всех терминалов в его шлюзовом поддереве не превосходит Q;

2) среди всех остовов, удовлетворяющих первому свойству, T имеет наименьшую стоимость.

Задача CMST возникает при проектировании сетей телекоммуникаций, гидравлических сетей и автомобильных дорог. Она является NP-трудной даже в том случае, если все $q_i = 1$ (см. [8]), в связи с чем большое внимание уделяется разработке эвристических и метаэвристических методов её решения. Краткий обзор существующих методов можно найти в [11].

В данной работе рассматриваются два метода построения окрестности, возникающей при применении алгоритмов локального поиска. На их основе разработаны алгоритмы решения задачи CMST, реализующие классический и модифицированный метод имитации отжига.

2. Общая схема метода имитации отжига

Метод *имитации отжига* (simulated annealing) строит последовательность планов оптимизационной задачи на минимум, начиная с начального плана x_0 и на t-й итерации (итерации нумеруются с нуля) переходя от плана x_t к плану x_{t+1}. На каждой из итераций метод действует следующим образом. Сначала явно или неявно для плана x_t строится так называемая *окрестность* $N(x_t)$ — дискретная случайная величина, задающая множество «соседних» к x_t планов и для каждого из соседних планов — вероятность его выбора. Затем, с учётом вероятностей выбора, из окрестности случайным образом выбирается план x_{new}. Пусть $f(x)$ — стоимость плана x. Если $f(x_{new}) < f(x_t)$, то в качестве x_{t+1} выбирается план x_{new}. Иначе x_{t+1} задаётся по правилу

$$x_{t+1} = \begin{cases} x_{new} \text{ с вероятностью } p_t \\ x_t \text{ с вероятностью } 1 - p_t \end{cases}$$

Здесь p_t — *вероятность перехода к худшему решению на t-й итерации* — некоторая функция от t, x_{new} и x_t.

Процесс построения последовательности планов задачи завершается после выполнения T итераций. Среди всех построенных планов x_i выбирается план с наименьшей стоимостью. Этот план и является результатом работы алгоритма, реализующего метод имитации отжига.

Центральный момент любого алгоритма, основанного на описанном методе — построение окрестности $N(x)$. Явное построение требует перечисления всех лежащих в $N(x)$ планов и для каждого — указания вероятности его выбора в качестве соседнего к плану x. Но поскольку метод имитации отжига не предполагает на каждой итерации перебора всех планов из окрестности, то, как правило, нет необходимости строить $N(x)$ явно. Вместо этого используется алгоритм случайной модификации плана x, а под $N(x)$ понимается множество всех планов, которые

могут быть получены из x применением этого алгоритма (каждый план из $N(x)$ с некоторой вероятностью возвращается алгоритмом).

3. Построение окрестности

В этом разделе мы кратко приведём некоторые алгоритмы модификации плана x, встречающиеся в литературе, посвящённой задаче CMST. Каждый из этих алгоритмов задаёт некоторую окрестность $N(x)$. Можно разделить эти алгоритмы на две группы, исходя из принципа их работы.

А) Алгоритмы, основанные на удалении/добавлении рёбер.

Алгоритм 1. (см. [7]) Добавим к дереву x случайное ребро, ранее не принадлежащее ему. В образовавшемся при этом цикле найдём максимальное ребро, удаление которого не нарушит ограничения на пропускную способность. Удалим это ребро.

Алгоритм 2. (см. [6]) Удалим из дерева x случайное ребро. Дерево разобьётся на две компоненты связности. Среди всех рёбер, концы которых лежат в разных компонентах, найдём минимальное ребро, добавление которого не нарушит ограничения на пропускную способность. Добавим это ребро.

Алгоритм 3. (см. [2]) Удалим из дерева x два случайных ребра. Найдём два ребра минимального суммарного веса, добавление которых восстанавливает целостность дерева и не нарушает ограничения на пропускную способность (при этом одно из этих рёбер может совпадать с одним из удалённых). Добавим эти два ребра.

В) Алгоритмы, основанные на переносе вершин.

Алгоритмы этой группы оперируют с каждым шлюзовым поддеревом как со множеством вершин, лежащих в этом поддереве. Стоимость такого поддерева определяется как стоимость минимального остова, построенного на множестве вершин поддерева и вершине v_0. Если степень v_0 в данном минимальном остове больше единицы, то шлюзовое поддерево разбивается на несколько поддеревьев, каждое — с корнем в своём шлюзе.

Алгоритм 4. (см. [4]) Выберем случайный терминал v_i. Найдём шлюзовое поддерево T_j, отличное от того, в котором лежит v_i, и такое, что сумма весов терминалов в T_j не превышает $Q - q_i$. Переместим v_i в T_j.

Алгоритм 5. (см. [4]) Выберем пару случайных терминалов v_i и v_j, лежащих в шлюзовых поддеревьях T_i и T_j, таких, что сумма весов терминалов в T_i не превышает $Q - q_j + q_i$, а сумма весов терминалов в T_j не превышает $Q - q_i + q_j$. Переместим v_i в T_j, а v_j — в T_i.

Алгоритм 6. (см. [9]) Выберем случайный терминал v_i. Найдём шлюзовое поддерево T_j, отличное от того, в котором лежит v_i, и такое, что сумма весов всех терминалов, лежащих в T_j либо являющихся потомками v_i в дереве x, не превышает Q (здесь и далее вершина v_i также считается своим потомком). Переместим всех потомков v_i в поддерево T_j либо создадим новое шлюзовое поддерево, состоящее из всех потомков v_i (другими словами, перевесим поддерево вершины v_i к корню дерева).

Алгоритм 7. (см. [3]) Рассмотрим множество терминалов $v_{p_1}, v_{p_2}, \ldots, v_{p_k}$. Пусть T_j — шлюзовое поддерево, в котором лежит терминал v_{p_j}. Также положим $v_{p_{k+1}} = v_{p_1}$, $T_{k+1} = T_1$. Если для любых $1 \leqslant i < j \leqslant k$ поддеревья T_i и T_j различаются и для любого $1 \leqslant i \leqslant k$ суммарный вес терминалов в T_{i+1} не превышает $Q - q_{p_i} + q_{p_{i+1}}$, переместим все вершины v_{p_i} в поддеревья T_{i+1}.

4. Реализация алгоритма

Остановимся на алгоритме 4 из предыдущего раздела (Amberg и др., 1996).

Будем обозначать $T[v]$ шлюзовое поддерево с корнем v. В качестве $N(x_t)$ возьмём множество планов, получаемых из x_t перемещением некоторого терминала v_s из шлюзового поддерева $T[v_{old}]$ в шлюзовое поддерево $T[v_{new}]$. Ниже приведён алгоритм выбора случайного плана x_{new} из $N(x_t)$.

1. Пусть $\tilde{V}$ — множество всех шлюзов в x_t. Выберем из $\tilde{V}$ случайный шлюз v_{old}.
2. Рассмотрим множество всех терминалов шлюзового поддерева $T[v_{old}]$. Выберем из них случайный терминал v_s. Удалим v_s из $T[v_{old}]$.
3. Пусть $\Omega = \{v_i \in \tilde{V} \setminus v_{old} \mid \sum\limits_{v_j \in T[v_i]} q_j \leq Q - q_s\} \cup v_s$. Выберем из Ω случайную вершину v_{new}.
4. Если $v_{new} = v_s$, добавим ребро $\{v_0, v_s\}$, создав тем самым новое шлюзовое поддерево $T[v_s]$.
5. Иначе добавим v_s в шлюзовое поддерево $T[v_{new}]$.
6. Искомый план является планом x_{new}.

При выборе случайного элемента из множества на шагах 1–3 будем считать все элементы данного множества равновероятными.

Оценим временную сложность описанного алгоритма. Пусть перед началом его работы поддерево $T[v_{old}]$ содержит n_{old} вершин, а поддерево $T[v_{new}]$ — n_{new} вершин. Выбор шлюзов на шагах 1 и 3 имеет сложность $\mathrm{O}(|\tilde{V}|)$, выбор терминала на шаге 2 — $\mathrm{O}(n_{old})$. Шаг 4 выполняется за $\mathrm{O}(1)$.

Осталось оценить сложность вычисления стоимости изменённых шлюзовых поддеревьев $T[v_{old}]$ и $T[v_{new}]$ на шагах 2 и 5. На шаге 2 требуется построить минимальный остов на множестве из n_{old} вершин: $n_{old}-1$ вершины поддерева $T[v_{old}]$ и корня v_0. Это может быть сделано за $\mathrm{O}(n_{old}^2)$ алгоритмом Прима [5]. На шаге 5 требуется построить минимальный остов на множестве из $n_{new}+2$ вершин: $n_{new}+1$ вершины поддерева $T[v_{new}]$ и корня v_0. Это можно сделать быстрее чем за $\mathrm{O}(n_{new}^2)$, если воспользоваться следующим утверждением.

Утверждение 1. *Любой минимальный остов T_1 полного графа на множестве вершин V можно превратить в некоторый минимальный остов T_2 полного графа на множестве вершин $V \cup v$, где $v \notin V$, добавив некоторые рёбра вида $\{u, v\}$, где $u \in V$, и, возможно, удалив некоторые рёбра T_1.*

Доказательство. Среди всех минимальных остовов полного графа на множестве вершин $V \cup v$ выберем остов, отличающийся от T_1 наименьшим количеством рёбер. Обозначим его T_2. Докажем, что T_2 не содержит рёбер вида $\{u_1, u_2\}$, где $u_1, u_2 \in V$ и $\{u_1, u_2\} \notin T_1$. От противного, пусть в T_2 есть такое ребро $\{u_1, u_2\}$. Удалим его из T_2, разбив T_2 на компоненты связности C_1 и C_2, $u_1 \in C_1$ и $u_2 \in C_2$. Рассмотрим простой путь из вершины u_1 в вершину u_2 по рёбрам дерева T_1. Найдём в этом пути ребро $\{u_i, u_j\}$ такое, что $u_i \in C_1$ и $u_j \in C_2$. Стоимость ребра $\{u_i, u_j\}$ не превосходит стоимости ребра $\{u_1, u_2\}$, поскольку T_1 — минимальный остов. Добавив в T_2 ребро $\{u_i, u_j\}$, получим противоречие с выбором остова T_2. Отсюда следует, что таких рёбер $\{u_1, u_2\}$ не существует, то есть любое ребро T_2 либо содержится в T_1, либо инцидентно вершине v. □

Из утверждения 1 следует, что стоимость изменённого шлюзового поддерева $T[v_{new}]$ на шаге 5 можно найти за время $\mathrm{O}(n_{new} \cdot \log(n_{new}))$ алгоритмом Краскала [5]. Следовательно, общая временная сложность описанного алгоритма — $\mathrm{O}(|\tilde{V}| + n_{old}^2 + n_{new} \cdot \log(n_{new}))$.

Приведённая оценка сложности показывает, что основное время уходит на удаление терминала из поддерева $T[v_{old}]$ с последующим перестроением минимального остова. Можно ускорить работу алгоритма, если потребовать, чтобы перемещаемый терминал v_s обязательно являлся листом поддерева $T[v_{old}]$. В самом деле, после удаления листа на шаге 2 оставшиеся в $T[v_{old}]$ рёбра уже будут образовывать минимальный остов, поэтому нет необходимости использовать алгоритм Прима. Сложность алгоритма уменьшается до $\mathrm{O}(|\tilde{V}| + n_{old} + n_{new} \cdot \log(n_{new}))$.

5. Модифицированный метод имитации отжига

Пусть алгоритм, реализующий классический метод имитации отжига, последовательно строил следующие планы оптимизационной задачи на минимум: x_k; $x_{k+1} = x_k$; $x_{k+2} = x_k$; $\ldots$; $x_{k+l} = x_k$; $x_{k+l+1} \neq x_k$. Допустим, $f(x_{k+l+1}) \geqslant f(x_k)$. Это означает, что на $(k+l)$-й итерации был осуществлён вероятностный переход к худшему решению. Заметим, что все планы x_{new}, найденные с k-й по $(k+l-1)$-ю итерацию, имели стоимость не меньше $f(x_k)$, но некоторые из них, вообще говоря, могли иметь стоимость меньше $f(x_{k+l+1})$. Это свойство специфично для классического метода имитации отжига.

Можно модифицировать метод имитации отжига так, чтобы в случае вероятностного перехода к худшему решению переход осуществлялся не к текущему соседу, а к лучшему из просмотренных ранее соседей (см. [1]). Полученный в результате такой модификации метод может быть применён к любой оптимизационной задаче, к которой применим классический метод.

Приведём формальное описание модифицированного метода.

1. Выберем некоторый план оптимизационной задачи в качестве начального. Обозначим его x_0. Положим $x^* = x_0$, $h^* = +\infty$.
2. Для t от 0 до $T-1$:
   a) Рассмотрим окрестность $N(x_t)$. Выберем из неё случайным образом план x_{new}. Положим $h = f(x_{new}) - f(x_t)$.
   b) Если $h < h^*$, положим $x^* = x_{new}$, $h^* = h$.
   c) Если $h < 0$, положим $x_{t+1} = x_{new}$, $h^* = +\infty$.
   d) Иначе с вероятностью p_t положим $x_{t+1} = x^*$, $h^* = +\infty$. С вероятностью $1 - p_t$ положим $x_{t+1} = x_t$.

Среди всех построенных планов x_i выбирается план с наименьшей стоимостью. Он является результатом работы алгоритма. Как и в случае классического метода имитации отжига, для создания алгоритма, укладывающегося в описанную схему, нужно зафиксировать три составляющие:

1) алгоритм построения начального плана x_0;
2) алгоритм построения окрестности $N(x_t)$;
3) функцию p_t.

6. Вычислительный эксперимент

Для сравнения качества различных точных, эвристических и метаэвристических алгоритмов разработаны специальные тестовые примеры CMST. Мы будем использовать в ходе вычислительного эксперимента тестовый набор `cm200rX-Y` из библиотеки OR-Library [12]. Примеры из этого набора были предложены Y. Sharaiha. Граф в них содержит 200

вершин ($n = 199$). Веса терминалов и стоимости всех рёбер — целые числа, имеющие равномерное распределение на отрезке [1,100]. Примеры не удовлетворяют неравенству треугольника. В имени примера `cm200rX-Y` число `X` обозначает номер серии (от 1 до 5), а число `Y` — пропускную способность Q (200, 400 или 800). Примеры из одной серии различаются только значением Q.

Мы реализовали алгоритмы с перемещением произвольного терминала и с перемещением листа, для каждого — классическую и модифицированную версию имитации отжига. В качестве x_0 было выбрано тривиальное решение «звезда», в котором каждый терминал является шлюзом. После этого производилось 100 миллионов итераций отжига. В качестве вероятности p_t перехода к худшему соседу мы использовали функцию вида:

$$p_t = \begin{cases} p, \text{ если } f(x_{new}) - f(x_t) \leqslant D \\ 0, \text{ если } f(x_{new}) - f(x_t) > D \end{cases}$$

где p и D — параметры алгоритма.

Каждая из программ была запущена 10 раз на каждом из 15 тестовых примеров. Тестирование производилось на персональном компьютере с процессором Intel Core2 Duo E4600 2.40 GHz.

Программы, реализующие перемещение произвольного терминала, запускались с параметрами $p = 0.003$ и $D = 5$, программы, реализующие перемещение листа — с параметрами $p = 0.005$ и $D = 20$.

В таблицах I и II приведены результаты работы программ: в Basic — реализующей классический метод имитации отжига, в Enhanced — реализующей модифицированный метод имитации отжига. Для каждого примера приведена стоимость лучшего известного решения (Best), указаны лучший (Min) и средний (Avg) результат работы программы на этом примере и среднее относительное отклонение найденного решения от оптимального (δ) в процентах. Лучшие известные решения для примеров взяты из работы [10].

Результаты эксперимента показывают, что на всех примерах модифицированная версия имитации отжига даёт значительно лучший результат, чем классическая, как для окрестности с перемещением произвольного терминала, так и для окрестности с перемещением листа. Если сравнивать алгоритмы построения окрестностей, то видно, что при применении модифицированного метода перемещение произвольного терминала даёт результаты лучше на всех примерах, кроме `cm200r2-400`, `cm200r4-200` и `cm200r5-400`, хотя это улучшение для большинства примеров не превышает 1%.

В таблице III отражено время работы программ. Время работы существенно зависит от значения Q (так как оно влияет на количество терминалов в шлюзовых поддеревьях), несущественно зависит от примера (время работы для примеров с одинаковыми Q отличается незна-

Таблица 1.

Перемещение произвольного терминала

Test	**Basic**			**Enhanced**			**Best**
	Min	**Avg**	$\delta, \%$	**Min**	**Avg**	$\delta, \%$	
cm200r1-200	1075	1133.4	14.02	1019	1028.8	3.50	994
cm200r1-400	415	428.6	9.62	400	405.1	3.61	391
cm200r1-800	265	268.5	5.71	257	258.8	1.89	254
cm200r2-200	1316	1392.9	17.25	1221	1242.9	4.62	1188
cm200r2-400	492	509.7	7.08	485	488.5	2.63	476
cm200r2-800	306	308.3	4.86	296	297.3	1.12	294
cm200r3-200	1474	1507.7	14.83	1358	1372.2	4.51	1313
cm200r3-400	583	591.8	5.87	565	568.2	1.65	559
cm200r3-800	373	374.9	3.85	362	364.5	0.97	361
cm200r4-200	1017	1081.4	17.93	945	962.7	4.98	917
cm200r4-400	409	416.5	7.07	395	398.7	2.49	389
cm200r4-800	283	287.5	4.55	277	278.4	1.24	275
cm200r5-200	1056	1096.3	15.64	966	989.5	4.38	948
cm200r5-400	444	452.2	8.18	429	433.7	3.76	418
cm200r5-800	304	305.9	4.76	295	295.8	1.30	292

чительно) и не зависит от того, классическая или модифицированная версия имитации отжига применяется. Отметим также, что применение алгоритма Краскала вместо алгоритма Прима, несмотря на выигрыш в асимптотике, не даёт реального ускорения на данных примерах, поскольку приходится строить минимальный остов для графа с малым числом вершин. Программы, реализующие перемещение листа, работают быстрее программ, реализующих перемещение произвольного терминала, за счёт того, что на каждой итерации ищется минимальный остов в одном шлюзовом поддереве, а не в двух. Эта разница наиболее ярко выражена на примерах с $Q = 800$, на которых основное время работы программ уходит на поиск минимального остова.

Список литературы

1. Ипатов А. В. Модифицированный метод имитации отжига в задаче CMST / А. В. Ипатов // Информ. бюл. Ассоциации мат. программирования. – Екатеринбург : УрО РАН, 2011. – Вып. 12. – С. 182–183.
2. Ипатов А. В. Об одном методе построения окрестности в задаче CMST / А. В. Ипатов // Современные проблемы математики : тез. 42-й всерос. молодёж. шк.-конф. – Екатеринбург : ИММ УрО РАН, 2011. – С. 161–163.
3. Ahuja R. K. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem / R. K. Ahuja, J. B. Orlin, D. Sharma // Mathematical Programming. – 2001. – Vol. 91. – P. 71–97.

Таблица 2.

Перемещение листа

Test	**Basic**			**Enhanced**			**Best**
	Min	**Avg**	$\delta, \%$	**Min**	**Avg**	$\delta, \%$	
cm200r1-200	1082	1092.2	9.88	1027	1034.7	4.09	994
cm200r1-400	431	433.9	10.97	402	408.7	4.53	391
cm200r1-800	266	268.8	5.83	259	262.3	3.27	254
cm200r2-200	1275	1308.6	10.15	1232	1248.8	5.12	1188
cm200r2-400	510	514.9	8.17	482	487.7	2.46	476
cm200r2-800	305	308.5	4.93	298	299.6	1.90	294
cm200r3-200	1420	1450.5	10.47	1375	1383.9	5.40	1313
cm200r3-400	592	594.9	6.42	566	572	2.33	559
cm200r3-800	372	375.9	4.13	366	367.7	1.86	361
cm200r4-200	980	1003.9	9.48	951	955.9	4.24	917
cm200r4-400	423	427.2	9.82	399	401.9	3.32	389
cm200r4-800	286	287.3	4.47	277	279.9	1.78	275
cm200r5-200	1038	1049.6	10.72	992	997.3	5.20	948
cm200r5-400	450	458.3	9.64	427	432.4	3.44	418
cm200r5-800	304	306	4.79	297	298.8	2.33	292

Таблица 3.

Время работы программ

Q	Перемещение терминала	Перемещение листа
200	109–121 сек	85–95 сек
400	235–244 сек	145–154 сек
800	617–633 сек	300–314 сек

4. Amberg A. Capacitated minimum spanning trees: Algorithms using intelligent search / A. Amberg, W. Domschke, S. Voss // Combinatorial Optimization: Theory and Practice. – 1996. – Vol. 1. – P. 9–39.
5. Introduction to algorithms, second edition / T. Cormen, C. Leiserson, R. Rivest, C. Stein. – The MIT Press, Cambridge, MA, 2001. – 1184 p.
6. Elias D. Topological design of multipoint teleprocessing networks / D. Elias, M. Ferguson // IEEE Transactions on Communications. – 1974. – Vol. 22. – P. 1753–1762.
7. Optimal design of centralized computer networks / H. Frank, I.T. Frisch, R. Van Slyke, W. S. Chou // Networks. – 1971. – Vol. 1. – P. 43–57.
8. Papadimitriou C. H. The complexity of the capacitated tree problem / C. H. Papadimitriou // Networks. – 1978. – Vol. 8. – P. 217–230.
9. A tabu search algorithm for the capacitated shortest spanning tree / Y. M. Sharaiha, M. Gendreau, G. Laporte, I. H. Osman // Networks. – 1997. – Vol. 29. – P. 161–171.

10. Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation / E. Uchoa [et al.] // Mathematical Programming: Series A and B. – 2007. – Vol. 112, Iss. 2. – P. 443–472.
11. Voss S. Capacitated minimum spanning trees / S. Voss // Encyclopedia of optimization / eds. C. A. Floudas, P. M. Pardalos. – Kluwer Academic Publishers, Doordrecht, 2001. – Vol. 6. – P. 225–235.
12. Capacitated minimal spanning tree [Electronic resource]. – URL: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capmstinfo.html

---

A. Ipatov

Building a capacitated minimum spanning tree using simulated annealing

Abstract. In this paper we consider capacitated minimum spanning tree problem (CMST) which is NP-hard. We have developed enhanced simulated annealing method, which allows getting better solutions for CMST than the classical one. Computational results on the benchmark instances are reported.

Keywords: capacitated minimum spanning tree; simulated annealing; metaheuristic; neighborhood

Ипатов Александр Владимирович, Уральский государственный университет им. А.М.Горького (sandro@acm.timus.ru)

Ipatov Aleksandr, Ural State University (sandro@acm.timus.ru)