УДК 51

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧИ ФОРМИРОВАНИЯ МАЛЫХ ГРУПП С УЧЕТОМ МЕЖЛИЧНОСТНЫХ ОТНОШЕНИЙ

Шадибекова А. Ш.,
научный руководитель д-р физ.-мат. наук Быкова В.В.
Сибирский федеральный университет

На современном этапе развития общества актуальной становится проблема подбора квалифицированных кадров, что обусловлено созданием новых компаний, развитием торговых сетей, повышением требований к специалистам и рядом других факторов. В данной работе исследуется задача формирования производственных групп, в которой максимизируется уровень комфортности отношений между специалистами и исключаются напряженные отношения.

Построим теоретико-графовую модель данной задачи. Определим граф G с множеством вершин $V = \{1, \dots, n\}$ и множеством ребер $E \subseteq \{(i,j) | \, i,j \in V, i \neq j\}$. Вершины графа G соответствуют претендентам на включение в производственную группу, а ребра – отношениям между ними, причем $E = E_1 \cup E_2$, где

1) E_1 – множество ребер, отражающих комфортные отношения между специалистами,

2) E_2 – множество ребер, отвечающих напряженным отношениям.

Окрасим ребра графа G в различные цвета: ребра из E_1 – в зеленый цвет, а из E_2 – в красный. Задача оптимизации заключается в отыскании $V' \subseteq V$ такого, что подграф G'графа G, порожденный подмножеством V', не содержит ребер из E_2 и включает максимальное число ребер из E_1.

Возможно ситуация, когда между двумя специалистами отношения не определены. На языке теоретико-графовой модели это отвечает отсутствию какого-либо ребра между парой соответствующих вершин графа G. Предполагаем, что граф G не обязательно связен, множество его вершин V может содержать изолированные вершины. Относительно множество ребер возможны следующие случаи:

$-\ E = \emptyset$. Это случай безреберного графа. Одним из допустимых решений задачи будет включение всех специалистов в группу, т. е. $V' = V$;

$-\ E_1 \neq \emptyset, E_2 = \emptyset$. В этом случае в графе только зеленые ребра и между любой парой специалистов комфортные отношения. Все вершины графа, инцидентные зеленым ребрам, должны быть включены в оптимальное решение;

$-\ E_1 = \emptyset, E_2 \neq \emptyset$. В данном случае в графе G только красные ребра. Допустимым решением задачи является некоторое независимое подмножество множества вершин графа G

Известно, что поставленная выше задача является *NP*-трудной. Поэтому интерес представляют эвристические алгоритмы, работающие полиноминальное время. Разработаны два эвристических алгоритма HA_1, HA_2.

Основная идея алгоритма HA_1 заключается в том, что сначала все вершины исходного графа G включаются в решение, затем выбирает вершина с наибольшим числом инцидентных ей красных ребер и исключается из него.

В эвристике HA_2 для каждой вершины подчитывается ее вес, равный количеству инцидентных ей зеленых ребер, уменьшенных на число инцидентных ей красных ребер. Сначала все вершины исходного графа G включаются в решение, затем выбирается вершина с наименьшим весом, у которой имеются инцидентные ей красные ребра, и исключается из него. Критерий остановки обеих алгоритмов – отсутствие красных ребер в решении.

Опишем эвристики по шагам. Алгоритм HA_1.

Шаг 0**.** G':=G.

Шаг 1. Для каждой вершины графа G':= (V', E') вычисляем количество инцидентных ей красных ребер, входящих в текущее решение. Переходим на шаг 2.

Шаг 2**.** Выбираем вершину $i \in V'$ с наибольшим числом инцидентных красных ребер. Если таких вершин нет, то процесс завершается, иначе переходим на следующий шаг.

Шаг 3. Если вершину i одна, то G':= $G' \backslash \{i\}$. Если вершин несколько, то для каждой из них подсчитываем количество инцидентных ей зеленых ребер, входящих в текущее решение. Выбираем вершину k с минимальным числом зеленых ребер и удаляем ее из решения: G':= $G' \backslash \{k\}$. Переходим на шаг 1.

Алгоритм HA_2.

Шаг 0**.** G':=G.

Шаг 1. Для каждой вершины графа G':= (V', E') вычисляем вес. Переходим на шаг 2.

Шаг 2**.** Выбираем вершину $i \in V'$ с наибольшим весом, у которой имеются инцидентные ей красные ребра. Если таких вершин нет, то процесс завершается, иначе переходим на следующий шаг.

Шаг 3. Если вершину i одна, то G':= $G' \backslash \{i\}$. Если вершин несколько, то для каждой из них находим количество инцидентных ей зеленых ребер, входящих в текущее решение. Выбираем вершину k с минимальным числом зеленых ребер и удаляем ее из решения: G':= $G' \backslash \{k\}$. Переходим на шаг 1.

Трудоемкости алгоритмов HA_1, HA_2 одинаковые и равны $O(n^3)$. Разработанные эвристики можно применять для поиска приближенных решений задачи.

К настоящему время разработана программа, реализующая эвристический алгоритм HA_1, на языке программирования Delphi. Программа включает в себя следующие основные модули:

– модуль AddVWithName (добавляет вершину);

– модуль AddE (добавляет ребро);

– модуль Deg (вычисляет степень вершины);

– модуль CountOfRedEdgesForV (находить число зеленых ребер, инцидентных заданной вершине);

– модуль CountOfRedEdgesForV (находить число красных ребер, инцидентных заданной вершине);

– модуль TGraf.DelV (удаляет вершину);

– модуль TGraf.DelE (удаляет ребро) и др.

Выполнена экспериментальная проверка программы.