

МЕТОД ВЕТВЕЙ И ГРАНИЦ ДЛЯ РЕШЕНИЯ ЗАДАЧИ О ЛИНЕЙНОМ ПОРЯДКЕ НА ВЗВЕШЕННЫХ ТУРНИРАХ

И. Шарон, О. Юдри

В турнире, дугам которого приписаны неотрицательные веса, требуется найти множество дуг с минимальным суммарным весом, смена ориентации которых преобразует исходный турнир в транзитивный. Для решения этой задачи предлагается новый вариант метода ветвей и границ. При вычислении нижних оценок целевой функции используется техника лагранжевых релаксаций. Верхние оценки находятся с помощью метода шума. Приводятся результаты численного эксперимента на турнирах, содержащих не более 100 вершин.

1. Постановка задачи

Ориентированный граф $T = (V, E)$ с множеством вершин $V = \{v_1, \dots, v_n\}$, $n \geq 3$, и множеством дуг E называется *турниром*, если в T любые две вершины $v_i, v_j \in V$ соединены только одной дугой (v_i, v_j) либо дугой (v_j, v_i) из E (основные определения и результаты о турнирах см. в [28]). Турнир $Q = (V, F)$ называется *транзитивным*, если из условия $(v_i, v_j), (v_j, v_k) \in F$ следует, что $(v_i, v_k) \in F$. Транзитивный турнир $Q = (V, F)$ эквивалентен линейному упорядочению вершин $R = (v_{i_1}, \dots, v_{i_n})$, в котором $v_i < v_j$, если $(v_i, v_j) \in F$.

Пусть каждой дуге $(v_i, v_j) \in E$ турнира $T = (V, E)$ приписан вес w_{ij} — произвольное неотрицательное число. Такой турнир называется *взвешенным*. *Удаленностью* взвешенного турнира T от взвешенного транзитивного турнира Q (или от соответствующего линейного порядка R) [4] называется величина

$$f(Q) = \sum_{(v_i, v_j) \in E \setminus F} w_{ij}, \quad (1)$$

равная сумме весов дуг из множества $E \setminus F$, смена ориентации которых преобразует турнир T в Q . Требуется найти

$$f^* = \min_Q f(Q), \quad (2)$$

где минимум берется по всем транзитивным турнирам Q .

Задача (2) о линейном порядке встречается в различных приложениях. Приведем три частных случая этой задачи, имеющих самостоятельный интерес.

Если вес каждой дуги турнира T равен 0 или 1, то получаем NP-трудную [19] задачу о множестве дуг, разрезающих контуры [8].

Если веса всех дуг турнира T равны 1, то получаем задачу Слейтера [31] о преобразовании турнира в линейный порядок. В этом случае минимальное значение f^* называется *индексом Слейтера* для турнира T и обозначается через $i(T)$. Вычислительная сложность задачи Слейтера неизвестна.

Третьим примером является задача Кемени о голосовании [20]. Обозначим через $V = \{v_1, \dots, v_n\}$ множество кандидатов на некоторый пост. Голосование проводят m избирателей. Для каждого избирателя известен линейный порядок его предпочтений на множестве кандидатов V . Пусть m_{ij} — число избирателей, предпочитающих кандидата v_j кандидату v_i . Дуга (v_i, v_j) принадлежит множеству дуг E турнира T , если $m_{ij} \geq m_{ji}$, и имеет вес $w_{ij} = m_{ij} - m_{ji}$. Величина $f(Q)$ интерпретируется как число разногласий между отдельными избирателями (турнир T) и коллективным предпочтением (турнир Q). Задача Кемени NP-трудна [17].

В силу сказанного выше задача (2) является NP-трудной. Для ее точного решения применяется метод ветвей и границ. Алгоритмом, предложенным в [22, 23], решались задачи на случайных турнирах при $n \leq 13$. В [25] этот алгоритм был улучшен, и размерность решаемых задач повысилась до $n = 17$. Алгоритмом из [32] решались задачи размерности $n = 25$. В [18] приведены результаты экспериментов на турнирах, взятых из практики, с 34 вершинами и на случайных турнирах с 25 вершинами. Используя методы линейного программирования, авторы работ [6] и [26] решали задачи с числом вершин 30 и 72 соответственно. Отметим работы, в которых применялись методы отсечения к задачам, имеющим размерность до 71 вершины [13] и до 80 вершин [15].

Алгоритм ветвей и границ, предлагаемый в настоящей работе, является модификацией подхода, рассмотренного в [3]. Алгоритм позволяет решать задачи достаточно большой размерности за приемлемое время. Вычисление нижних оценок функционала основано на лагранжевых релаксациях. Верхние оценки рассчитываются с помощью метода шума, который очень часто находит оптимальное решение в задачах небольшой размерности.

2. Нижняя оценка

Введем множества $P = \{(i, j) \mid 1 \leq i < j \leq n\}$ и $U = \{(i, j, k) \mid 1 \leq i < j < k \leq n\}$. Турниру $Q = (V, F)$ поставим в соответствие вектор

$X = (x_{ij}), (i, j) \in P$, с компонентами $x_{ij} = 1$, если $(v_i, v_j) \in F$, и $x_{ij} = 0$, если $(v_j, v_i) \in F$. Доопределим множество весов турнира $T = (V, E)$, полагая $w_{ij} = 0$, если $(v_i, v_j) \notin E$, и рассмотрим функцию

$$g(X) = \sum_{(i,j) \in P} (w_{ij}(1 - x_{ij}) + w_{ji}x_{ij}).$$

Очевидно, что $f(Q) = g(X)$. Поэтому задача (2) эквивалентна следующей: найти

$$f^* = \min_X g(X) \quad (3)$$

при ограничениях

$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1, \quad (i, j, k) \in U, \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad (i, j) \in P. \quad (5)$$

Неравенства (4) соответствуют условию транзитивности турнира Q .

Обозначим через $\Lambda = (\lambda_{ijk})$ и $M = (\mu_{ijk})$, где $(i, j, k) \in U$, произвольные векторы с неотрицательными компонентами. Выпишем функцию Лагранжа задачи (3)–(5):

$$\begin{aligned} L(X, \Lambda, M) = g(X) + \sum_{(i,j,k) \in U} \lambda_{ijk}(x_{ij} + x_{jk} - x_{ik} - 1) \\ - \sum_{(i,j,k) \in U} \mu_{ijk}(x_{ij} + x_{jk} - x_{ik}). \end{aligned}$$

После простых преобразований получаем

$$L(X, \Lambda, M) = \sum_{(i,j) \in P} a_{ij}x_{ij} - \sum_{(i,j,k) \in U} \lambda_{ijk} + C. \quad (6)$$

Здесь для краткости записи введено обозначение

$$\begin{aligned} a_{ij} = w_{ji} - w_{ij} + \sum_{l=j+1}^n (\lambda_{ijl} - \mu_{ijl}) + \sum_{l=1}^{i-1} (\lambda_{lij} - \mu_{lij}) \\ - \sum_{l=i+1}^{j-1} (\lambda_{ilj} - \mu_{ilj}), \end{aligned}$$

а через C обозначены слагаемые, не зависящие от X, Λ и M .

Нетрудно убедиться в том, что $g(X) \geq L(X, \Lambda, M)$ при любом векторе X , удовлетворяющем (4) и (5). Поэтому

$$f^* \geq \min_X L(X, \Lambda, M), \quad (7)$$

где минимум берется при ограничениях (5) и какой-либо части ограничений (4). Если учитывать только условия (5), то минимум в (7) достигается на векторе $\hat{X} = (\hat{x}_{ij})$ с компонентами $\hat{x}_{ij} = 1$, если $a_{ij} < 0$, и $\hat{x}_{ij} = 0$ в противном случае. В результате получаем нижнюю оценку оптимума

$$f^* \geq L(\hat{X}, \Lambda, M). \quad (8)$$

Естественно выбирать векторы Λ и M , стремясь максимизировать функцию $L(\hat{X}, \Lambda, M)$. Это можно сделать, например, с помощью следующей процедуры [27]. Начиная с произвольно выбранных векторов $\Lambda^0 = (\lambda_{ijk}^0)$ и $M^0 = (\mu_{ijk}^0)$, строится последовательность векторов $\Lambda^t = (\lambda_{ijk}^t)$ и $M^t = (\mu_{ijk}^t)$, $t = 1, \dots, I$, по формулам [2]

$$\begin{aligned} \lambda_{ijk}^t &= \max\{0, \lambda_{ijk}^{t-1} + \gamma(\hat{x}_{ij} + \hat{x}_{jk} - \hat{x}_{ik} - 1)\}, \\ \mu_{ijk}^t &= \max\{0, \mu_{ijk}^{t-1} - \gamma(\hat{x}_{ij} + \hat{x}_{jk} - \hat{x}_{ik})\}. \end{aligned}$$

Размер шага γ определяется выражением

$$\gamma = \frac{\rho(\hat{g} - L(\hat{X}, \Lambda^{t-1}, M^{t-1}))}{n(n-1)(n-2)}.$$

Здесь \hat{g} — наименьшее из значений $g(\hat{X})$, полученных на предыдущих шагах. На каждом шаге параметр ρ умножается на заданную величину $\theta < 1$.

Если начальное значение ρ выбрать слишком большим, то на некотором шаге может выполняться неравенство

$$L(\hat{X}, \Lambda^t, M^t) < L(\hat{X}, \Lambda^{t-1}, M^{t-1}).$$

В этом случае будем говорить, что алгоритм попал в состояние *дрейфа*. Если начальное значение ρ слишком мало, то сходимость алгоритма замедляется. Поэтому начальное значение ρ следует выбирать достаточно большим, а в случае возникновения дрейфа дополнительно умножать его на некоторый коэффициент $\eta < 1$.

Пусть Λ и M — некоторые множители Лагранжа, например полученные в результате описанной выше процедуры. Оценку (8) можно улучшить, учитывая при взятии минимума в (7) не только (5), но и неравенства (4). Выберем подмножество S из множества троек U , обладающее свойствами:

- 1) для любых двух троек (i, j, k) и (i', j', k') из S все пары (i, j) , (j, k) , (i, k) , (i', j') , (j', k') , (i', k') различны;
- 2) для любой тройки $(i, j, k) \in S$ выполняются либо неравенства

$$a_{ij} > 0, \quad a_{jk} > 0, \quad a_{ik} < 0, \quad (9)$$

либо неравенства

$$a_{ij} < 0, \quad a_{jk} < 0, \quad a_{ik} > 0. \quad (10)$$

Введем обозначение множества пар

$$B = \bigcup_{(i,j,k) \in S} \{(i,j), (j,k), (i,k)\}$$

и запишем (6) в следующем виде:

$$L(X, \Lambda, M) = \sum_{(i,j) \in P \setminus B} a_{ij} x_{ij} + \sum_{(i,j,k) \in S} (a_{ij} x_{ij} + a_{jk} x_{jk} + a_{ik} x_{ik}) - \sum_{(i,j,k) \in U} \lambda_{ijk} + C.$$

Вычисляя минимум в (7) с учетом ограничений (5), неравенств (4) при $(i,j,k) \in S$ и принимая во внимание (9) и (10), нетрудно убедиться в том, что

$$f^* \geq L(\hat{X}, \Lambda, M) + \sum_{(i,j,k) \in S} \min\{|a_{ij}|, |a_{jk}|, |a_{ik}|\}. \quad (11)$$

Слагаемые последней суммы назовем *штрафами*. Полученная оценка (11) лучше (8), если $S \neq \emptyset$.

Множество S можно строить, например, следующим способом. Из множества U выбирается тройка (i,j,k) , удовлетворяющая условию 2 и имеющая максимальный штраф

$$\min\{|a_{ij}|, |a_{jk}|, |a_{ik}|\}. \quad (12)$$

Она включается в S и удаляется из U . Далее процесс повторяется. На очередном шаге из оставшихся троек находится тройка (i,j,k) , удовлетворяющая условиям 2 и 1 вкпе с ранее выбранными тройками и имеющая максимальный штраф (12). Она добавляется в S и удаляется из U .

3. Верхняя оценка

Рассмотрим следующую задачу. Заданы конечное множество Z и функция $h(z)$, $z \in Z$. Требуется найти

$$h^* = \min_{z \in Z} h(z). \quad (13)$$

В основе многих подходов [1, 14, 24, 29] приближенного решения задачи (13) лежит метод спуска. Для любого элемента $z \in Z$ вводится подмножество элементов $\Gamma(z) \subset Z$, называемых *соседними* с z . Начиная с произвольного элемента $z_0 \in Z$, строится последовательность z_1, \dots, z_m со следующими свойствами:

$$\begin{aligned} z_t &\in \Gamma(z_{t-1}), \quad h(z_t) = \min\{h(z) \mid z \in \Gamma(z_{t-1})\}, \\ h(z_t) &< h(z_{t-1}), \quad t = 1, \dots, m. \end{aligned}$$

Результатом этого процесса является элемент z_m , наилучший из рассмотренных.

Используемый в данной работе метод шума [9] работает по тому же принципу, если добавить к целевой функции случайную величину, распределенную на отрезке $[-a, a]$. Эта случайная величина называется *шумом*, а число a — *уровнем* шума. В процессе работы метода уровень шума уменьшается до нуля.

Вернемся к задаче (2). Вместо транзитивных турниров Q будем говорить о соответствующих линейных порядках R на множестве вершин V и писать $f(R)$ вместо $f(Q)$. Пусть $R = (v_{i_1}, \dots, v_{i_n})$ — произвольный порядок и p, q — некоторые его позиции. Порядок R' , получаемый из R перемещением вершины v_{i_p} из позиции p в позицию q и сдвигом остальных вершин, назовем *соседним* с R . Метод спуска может быть применен к задаче (2), например, следующим образом. Алгоритм начинает работу с произвольного порядка. Последовательно вычисляется наилучшая позиция для вершины v_1 , затем для v_2 и так далее до v_n , после чего процесс продолжается, вновь начиная с вершины v_1 .

Перейдем к описанию метода шума применительно к задаче (2). Обозначим через $\Delta f(R, R') = f(R') - f(R)$ приращение функции (1), получаемое при переходе от порядка R к соседнему с ним порядку R' , и введем «возмущенное» приращение

$$\Delta f_{\xi}(R, R') = \Delta f(R, R') + \sum_{k=1}^{|p-q|} \xi_k \max_{(v_i, v_j) \in E} w_{ij},$$

где ξ_k — шумовая добавка с уровнем a — случайная величина, распределенная равномерно на отрезке $[-a, a]$. Алгоритм метода шума находит приближенное решение \hat{R} задачи (2) по следующей схеме [10, 11].

Шаг 1. Выбрать произвольный порядок R и положить $\hat{R} = R$, $a = A$, $t = 1$.

Шаг 2. Выбрать любой порядок R' , соседний с R , и вычислить приращение $\Delta f_{\xi}(R, R')$.

Шаг 3. Если $\Delta f_{\xi}(R, R') < 0$, то положить $R = R'$. Если $f(R) < f(\hat{R})$, то положить $\hat{R} = R$.

Шаг 4. Если $t \equiv 0 \pmod{t_1}$, то уменьшить уровень шума a .

Если $t \equiv 0 \pmod{t_2}$, то применить к R алгоритм метода спуска. Получаемый при этом порядок вновь обозначается через R , и если $f(R) < f(\hat{R})$, то полагается $\hat{R} = R$.

Если $t \equiv 0 \pmod{t_3}$, то положить $R = \hat{R}$.

Шаг 5. Если $t < t_0$, то положить $t = t + 1$ и вернуться к шагу 2. В противном случае закончить работу.

В этой схеме величина A обозначает максимальный уровень шума, t_0 — общее число итераций, t_1 — число итераций с текущим уровнем шума, t_2 — число итераций между двумя последовательными применениями метода спуска, t_3 — число итераций, через которое происходит замена текущего решения на наилучшее найденное.

4. Метод ветвей и границ

Корневой вершине дерева поиска поставим в соответствие множество всех линейных порядков. Алгоритм начинает работу с просмотра корневой вершины. С помощью метода шума вычисляется приближенное решение \hat{R} . Наилучшее текущее решение, найденное алгоритмом, назовем *рекордом* и обозначим через R^* . Сначала полагается $R^* = \hat{R}$.

Некорневой вершине N дерева поиска ставятся в соответствие все линейные порядки, начинающиеся с заданной последовательности v_{i_1}, \dots, v_{i_p} . Эта последовательность называется *частичным решением*. *Предком* вершины N в дереве поиска называется вершина, содержащая все линейные порядки, которые начинаются с $v_{i_1}, \dots, v_{i_{p-1}}$, а *потомком* вершины N — вершина, содержащая все линейные порядки с началом $v_{i_1}, \dots, v_{i_p}, v_{i_{p+1}}$, где $v_{i_{p+1}} \in V \setminus \{v_{i_1}, \dots, v_{i_p}\}$. В дальнейшем используются обозначения:

$BS = (v_{i_1}, \dots, v_{i_p})$ — частичное решение,

$S = \{v_{i_1}, \dots, v_{i_p}\}$ — соответствующее ему множество вершин турнира,

$OS = V \setminus S$ — дополнение множества S ,

T_{OS} — турнир, являющийся подграфом исходного турнира $T = (V, E)$ с множеством вершин OS .

(Обозначения BS и OS — аббревиатуры выражений «beginning section» и «out of section».)

Пусть BS — частичное решение, $Q = (S, F)$ — соответствующий транзитивный турнир и H — объединение множеств дуг F и $\{(u, v) \mid u \in S, v \in OS\}$. Тогда величина

$$W(BS) = \sum_{(v_i, v_j) \in H \setminus E} w_{ji} \quad (14)$$

равна сумме весов дуг из E , смена ориентации которых заведомо необходима для получения любого линейного порядка R , начинающегося с BS . Следовательно, $f(R) \geq W(BS)$.

Величина $W(BS)$ использовалась в [3] как нижняя оценка целевой функции. В [8] эта оценка усилена вследствие рассмотрения подтурниров на вершинах OS . Особо отметим задачу Слейтера (см. п. 1), когда все веса турнира T равны 1. Пусть s_1, \dots, s_n — полустепени исхода

вершин турнира T , упорядоченные по неубыванию, т. е. $s_1 \leq \dots \leq s_n$. Следуя работе [7], введем параметр

$$\sigma(T) = \sum_{j=1}^n |s_j - j + 1|/2. \quad (15)$$

Для индекса Слейтера $i(T)$ справедливо неравенство $i(T) \geq \sigma(T)$. Поэтому величина $W(BS) + \sigma(T_{OS})$ является нижней оценкой в задаче Слейтера, что лучше оценки (14) в общей ситуации.

Просмотр вершины N дерева поиска состоит в нахождении наилучшего порядка, который начинается с соответствующего частичного решения. Ниже приводится описание этой процедуры.

4.1. Дерево частичных решений

В [16] предложено хранить информацию о дереве поиска с помощью лексикографически упорядоченного *дерева частичных решений*. Каждой вершине этого дерева ставятся в соответствие все уже просмотренные частичные решения BS , имеющие одно и то же множество S . Через $W(S)$ обозначается наименьшее из значений $W(BS)$ среди таких BS .

Пусть N — очередная просматриваемая вершина дерева поиска, BS — соответствующее ей частичное решение с множеством элементов S . Пусть множество S уже встречалось в дереве частичных решений. Тогда если $W(BS) \geq W(S)$, то просмотр вершины N завершается, поскольку она заведомо не содержит решений, лучше ранее найденных. Число вершин, отсеченных таким образом, обозначим через *BegSec*. Если $W(BS) < W(S)$, то полагается $W(S) = W(BS)$. Если же рассматриваемое множество S не содержится в дереве частичных решений, то к этому дереву добавляется новая вершина, соответствующая множеству S , и полагается $W(S) = W(BS)$.

4.2. Вычисление верхней оценки

Пусть N — очередная просматриваемая вершина дерева поиска, BS — соответствующее ей частичное решение с множеством элементов S . Если $|S| = n - 1$, то вершина N содержит только один линейный порядок R . В случае $f(R) < f(R^*)$ текущий рекорд R^* заменяется на R . Просмотр вершины N завершен.

Если $|S| < n - 1$, то к турниру T_{OS} применяется алгоритм метода спуска. Пусть R' — линейный порядок на множестве вершин OS , найденный этим алгоритмом, и $R = BS \cup R'$. Тогда в случае $f(R) < f(R^*)$ полагается $R^* = R$.

Численные эксперименты показали, что метод шума почти всегда находит оптимальное решение в корневой вершине дерева поиска. Применение в других вершинах более сложных алгоритмов, чем метод спуска, приводит к увеличению общего времени счета.

4.3. Применение лагранжевой релаксации

Обозначим через $L(OS)$ нижнюю оценку, полученную с применением метода лагранжевой релаксации (см. п. 2) к турниру T_{OS} . Если $W(S) + L(OS) \geq f(R^*)$, то текущая вершина N дерева поиска *отсекается*, т. е. она не содержит решений лучше рекорда, и ее просмотр завершен. Число вершин, отсеченных с помощью лагранжевых релаксаций, обозначим через *relax*.

В корневой вершине дерева поиска начальные множители Лагранжа Λ и M полагаются равными нулю. В остальных вершинах в качестве начальных используются множители, полученные для предшествующей вершины.

Улучшение нижней оценки, рассмотренное в п. 2, с помощью вычисления штрафов может потребовать значительных затрат машинного времени. Эксперименты показали, что штрафы лучше использовать на первой итерации расчета множителей Лагранжа. На последующих итерациях штрафы, как правило, слишком малы, чтобы заметно улучшить нижнюю оценку.

Нижняя оценка для турнира T_{OS} сохраняется в дереве частичных решений. Она используется перед применением метода релаксации к очередной вершине N дерева поиска, имеющей то же самое множество S в частичном решении. Если полученная нижняя оценка не позволяет отсечь текущую вершину N , то совершается переход к процедуре ветвления.

4.4. Ветвление дерева поиска

Пусть v_{i_1}, \dots, v_{i_p} — текущее частичное решение и вершина v добавляется в конец этой последовательности. Ниже описываются процедуры, которые позволяют избежать просмотра нового частичного решения $BS = (v_{i_1}, \dots, v_{i_p}, v)$. Это удастся сделать, если

- нет ни одного оптимального порядка, начинающегося с BS ;
- для каждого порядка, начинающегося с BS , существует другой порядок с тем же значением целевой функции, но с лексикографически меньшим частичным решением. В этом случае лексикографически меньшее частичное решение будет просмотрено или уже просмотрено в другом месте дерева поиска.

Более точно, просматриваются все частичные решения, полученные сдвигом внутри BS некоторого интервала, содержащего v :

$$v_{i_1}, v_{i_2}, \dots, v_{i_{j-1}}, v_{i_{k+1}}, v_{i_{k+2}}, \dots, v_{i_p}, v, v_{i_j}, v_{i_{j+1}}, \dots, v_{i_k}, \quad 1 \leq j < k \leq p.$$

Если хотя бы одно из этих частичных решений лучше BS или имеет то же значение $W(BS)$, но является лексикографически меньшим, то частичное решение BS может быть отброшено. Обозначим через lex число вершин дерева поиска, исключенных по второму критерию. Число вершин, отброшенных по первому критерию (кроме случая $j = k = p$), обозначим через S_{moves} .

Если $j = k$, то вершина v_{i_j} перемещается непосредственно за вершину v . При $k = p$ вершина v ставится перед v_{i_p} . В случае $j = k = p$ вершины v и v_{i_p} меняются местами. Это приводит к обобщению процедуры, предложенной в [30] для турниров T с единичными весами, поскольку если v_{i_1}, \dots, v_{i_n} — порядок Слейтера в турнире T , то v_{i_1}, \dots, v_{i_n} — гамильтонов путь в T . Число вершин дерева поиска, отсеженных в случае $j = k = p$, обозначим через ham .

Рассмотрим порядок, начинающийся с частичного порядка BS . Выберем в нем некоторый интервал $v_{i_j}, v_{i_{j+1}}, \dots, v_{i_p}, v$, где $1 \leq j \leq p$, и поставим его после вершин из OS . Если значение целевой функции на таком порядке оказывается меньше текущей оценки, то рассматриваемая вершина дерева поиска отсекается. Число вершин, исключенных таким образом, обозначим через OS_{moves} . Описанная процедура основана на обобщении результата работы [5]: если v — первая вершина порядка Слейтера для турнира T , то число дуг, выходящих из v , не менее $(n-1)/2$.

Если частичное решение BS не было исключено ни одной из вышеперечисленных процедур, то в дереве поиска создается новая вершина. Этот процесс продолжается до тех пор, пока все вершины не будут либо просмотрены, либо отсежены.

5. Выбор параметров

В описанных выше методах вычисления нижних и верхних оценок целевой функции (1) присутствуют числовые параметры, от правильного выбора которых существенно зависят эффективность этих оценок и время работы алгоритма. Ниже приводятся значения параметров, использованные в численном эксперименте. Одни из параметров являются постоянными величинами. Другие рассчитываются в зависимости от числа вершин турнира $T = (V, E)$, его веса и индекса транзитивности, определяемого следующим образом [21]. Введем обозначения:

s_i — полустепень исхода вершины $v_i \in V$ в турнире T ,

$\alpha(T)$ — число контуров на трех вершинах в турнире T ,
 $\beta(n)$ — максимально возможное число контуров на трех вершинах среди всех турниров с n вершинами.

Известно [28], что

$$\alpha(T) = \frac{n(n-1)(n-2)}{6} - \sum_{i=1}^n \frac{s_i(s_i-1)}{2}.$$

Величина $\beta(n)$ равна $(n^3-4n)/24$, если n четно, и равна $(n^3-n)/24$, если n нечетно. *Индексом транзитивности* турнира T называется число

$$\tau(T) = 1 - \alpha(T)/\beta(n).$$

Индекс $\tau(T)$ транзитивного турнира T равен 1. Если в турнире T число вершин n нечетно и $s_i = (n-1)/2, i = 1, \dots, n$, то $\tau(T) = 0$. Индекс транзитивности $\tau(T)$ турнира T также равен нулю, если число вершин n четно и одна половина вершин имеет полустепени исхода, равные $n/2$, а другая половина — $(n-2)/2$. Как показали численные эксперименты, наиболее сложные задачи имеют малый индекс транзитивности. Нулевой индекс соответствует самому трудному случаю.

При вычислении верхних оценок методом шума были использованы значения параметров, предложенные в [19]. Максимальный уровень шума выбирался в зависимости от числа вершин n турнира T , его веса и индекса транзитивности $\tau(T)$. Уровень шума в конце работы алгоритма полагался равным нулю. Методу шума отводилось время счета, равное $10^{-4}n^3(1 + \varepsilon - \tau(T))$ секундам. Величина $\varepsilon = 10^{-5}$ добавлялась только в случае транзитивного турнира. Размер окрестностей, используемых в методе спуска, равен по порядку n^2 . Поэтому для турнира T , названного Judges100 (см. п. 6.1), с числом вершин $n = 100$ и индексом транзитивности $\tau(T) = 0,8275$ метод шума затрачивает примерно 17 секунд, что составляет менее 0,7% времени счета. С учетом этого задавалось общее число итераций t_0 метода шума. Остальные параметры выбирались согласно [11, 12]:

$$t_1 = t_2 = 4n(n-1), \quad t_3 = \left\lfloor \sqrt{n(n-1)t_0} \right\rfloor.$$

Расчет множителей Лагранжа, требуемых при вычислении нижних оценок, зависит от параметров I, ρ, θ и η (см. п. 2). Число итераций I выбиралось равным 1000 в корневой вершине дерева поиска и равным 20 в остальных вершинах. Коэффициент θ рассчитывался по формуле $\theta = 1 - 0,1/I$, т. е. $\theta = 0,9999$ в корневой вершине и $\theta = 0,995$ в других вершинах.

Обозначим через $\varphi(\tau), \tau \in [0, 1]$, кусочно-линейную функцию с узловыми значениями $\varphi(\tau) = 6, 66, 180, 480, 1200$ при значениях аргумента

$\tau = 0, 0,5, 0,8, 0,9, 1$ соответственно. В зависимости от индекса транзитивности $\tau(T)$ турнира T начальное значение параметра ρ полагалось равным $\varphi(\tau(T))$ в корневой вершине дерева поиска и равным $\varphi(\tau(T))/2$ в остальных вершинах.

Если на некотором шаге расчета множителей Лагранжа выполнялось неравенство

$$L(\hat{X}, \Lambda^t, M^t) < 0,98 \cdot L(\hat{X}, \Lambda^{t-1}, M^{t-1}),$$

то объявлялось состояние дрейфа и параметр ρ дополнительно умножался на коэффициент $\eta = 0,8$ в случае корневой вершины дерева поиска и $\eta = 0,9$ для некорневых вершин.

6. Численные эксперименты

Приведем результаты двух численных экспериментов. В первом эксперименте исследовалось число вершин дерева поиска, отсекаемых алгоритмом ветвей и границ, во втором оценивалось время работы алгоритма. Эксперименты проводились на ЭВМ SUN Sparc.

6.1. Отсечение вершин дерева поиска

Рассматривались три задачи: Slater32, Median39 и Judges100. В таблице приведены их характеристики: n — число вершин турнира, w — максимальный вес дуги, τ — индекс транзитивности турнира, $t_{\text{сч}}$ — время счета (в секундах).

| Задача | n | w | τ | $t_{\text{сч}}$ |
|-----------|-----|-----|--------|-----------------|
| Slater32 | 32 | 1 | 0,0875 | 509 |
| Median39 | 39 | 10 | 0,0785 | 2208 |
| Judges100 | 100 | 44 | 0,8275 | 2451 |

В задаче Slater32 для каждой пары вершин $\{v_i, v_j\}$, $i < j$, ориентация соединяющей их дуги выбирается случайным образом с вероятностью 0,5. Веса всех дуг равны 1.

В задаче Median39 турнир T строится тем же способом. Вес дуги — случайная величина, распределенная равномерно на отрезке $[1, 10]$.

Задача Judges100 является задачей Кемени (см. п. 1) с числом избирателей $m = 50$. Линейные порядки предпочтений избирателей строятся случайным образом. Вероятность предпочтения отдельным избирателем кандидата v_j кандидату v_i полагается равной $0,5 + 0,35(i - j)/(n - 1)$.

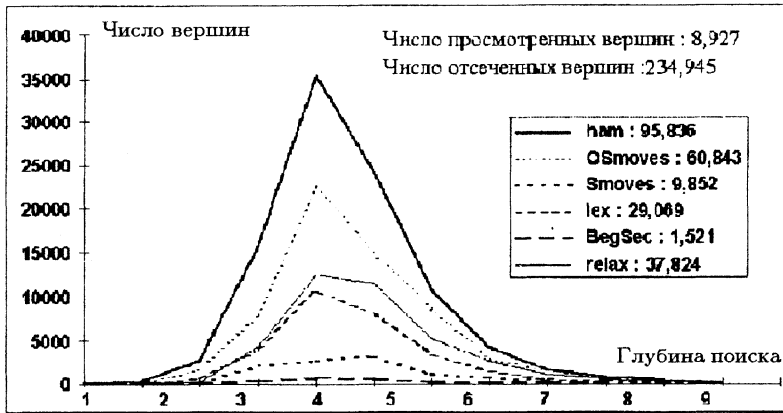


Рис. 1. Число отсеченных вершин в задаче Slater32

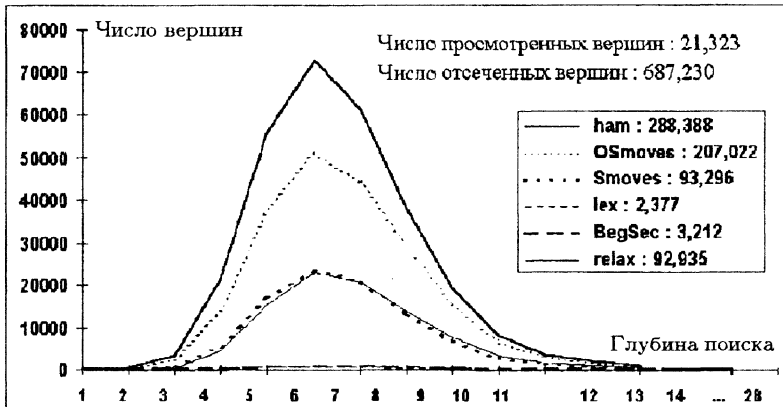


Рис. 2. Число отсеченных вершин в задаче Median39



Рис. 3. Число отсеченных вершин в задаче Judges100

При проведении численных расчетов подсчитывалось число вершин, отсеженных той или иной процедурой, на каждом уровне глубины дерева поиска. Результаты эксперимента изображены на рис. 1–3.

Вид приведенных графиков зависит от последовательности применения процедур алгоритма. Первая процедура использует гамильтоновы пути. Величина *ham* равна числу вершин, отсеженных этой процедурой. Далее следуют процедуры OS_{moves} , S_{moves} , *lex* и *BegSec*. Последней применяется лагранжева релаксация, которой соответствует величина *relax*. Изменение этого порядка может привести к другим результатам.

Анализ графиков показывает, что наиболее эффективны процедуры *ham*, OS_{moves} и *relax*. В задаче Median39 значительное число вершин отсекалось процедурой S_{moves} , а в задаче Slater32 — процедурой *lex*. Процедура *BegSec* использовалась редко, но ее применение существенно сокращало время счета.

Использование величины (15) при решении задачи Слейтера не дало хороших результатов. Исключение составляли почти транзитивные турниры и один специальный турнир с 200 вершинами и малым индексом транзитивности. В этих случаях сразу была отсежена корневая вершина.

6.2. Вычислительное время

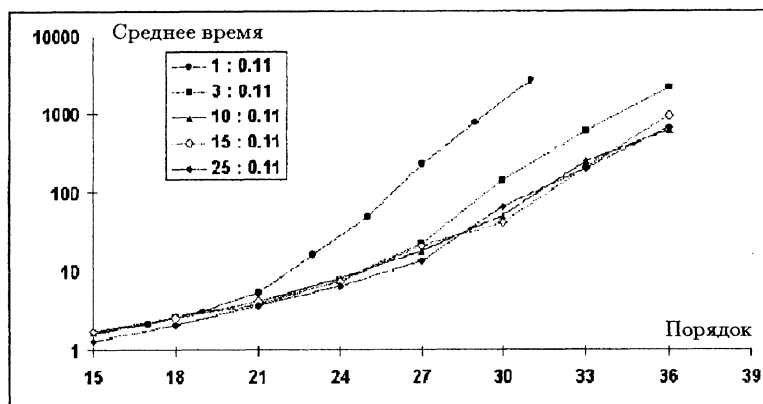
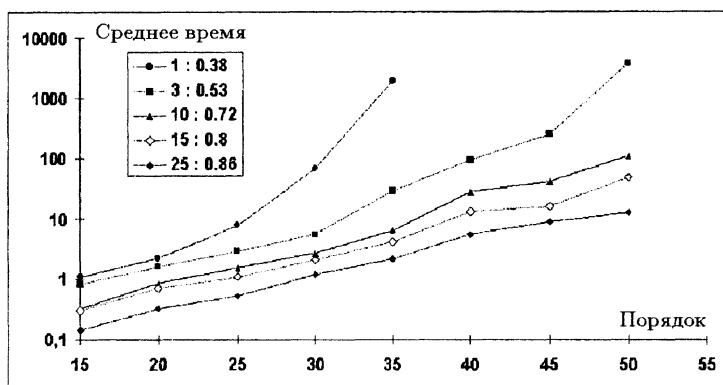
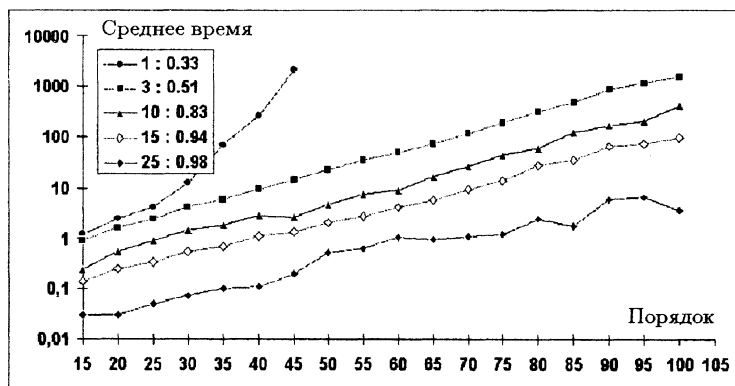
В данном разделе приводятся результаты для 5790 турниров с различными характеристиками. С каждым набором характеристик генерировалось 30 случайных турниров и вычислялось среднее время решения задачи. Все рассмотренные турниры порождены бинарными отношениями, которые задаются с помощью определенного числа избирателей.

Для первых трех видов турниров бинарные отношения предполагаются полными и асимметричными. Асимметричность означает, что v_i предпочитается тогда и только тогда, когда не предпочитается v_j . Для каждой пары вершин $\{v_i, v_j\}$, $1 \leq i < j \leq n$, избиратель решает вопрос о предпочтении в пользу v_i с вероятностью

$$\pi_1 + (\pi_2 - \pi_1)(j - i)/(n - 1), \quad 0,5 \leq \pi_1 \leq \pi_2 \leq 1.$$

Если $\pi_1 = \pi_2 = 0,5$, то получаем турнир, в котором дуга между v_i и v_j с равной вероятностью может быть как прямой, так и обратной. Обычно это приводит к трудным турнирам с низкими индексами транзитивности. Задачу с параметрами π_1, π_2 обозначим через $P(\pi_1; \pi_2)$.

Для первых трех видов задач число избирателей выбиралось равным 1, 3, 10, 15 и 25. Масштаб вертикальной оси, отражающей среднее время, выбран логарифмическим. В легендах приводится число избирателей и среднее значение индекса транзитивности. С увеличением размерности

Рис. 4. Время решения задач $P(0, 5; 0, 5)$ Рис. 5. Время решения задач $P(0, 5; 1)$ Рис. 6. Время решения задач $P(0, 75; 0, 75)$

индекс транзитивности в среднем уменьшается. В приведенных экспериментах это относительное изменение составляло несколько процентов (рис. 4–6).

Отметим, что в задаче $P(0, 5; 1)$ турниры получаются более трудными в вычислительном отношении, чем в задаче $P(0, 75; 0, 75)$. Еще более трудными получаются задачи $P(0, 5; 0, 5)$. Эксперименты показывают, что наиболее трудным задачам соответствуют турниры с низкими значениями индекса транзитивности. Для заданных π_1 и π_2 задача $P(\pi_1; \pi_2)$ труднее в случае одного избирателя (задача Слейтера) и легче, когда число избирателей увеличивается. Особенно хорошо это видно на задачах $P(0, 5; 1)$ и $P(0, 75; 0, 75)$. Для задач $P(0, 5; 0, 5)$ это, вероятно, следует из того факта, что веса все более различаются с ростом числа избирателей. Для задач $P(0, 5; 1)$ и $P(0, 75; 0, 75)$ с увеличением числа избирателей турниры становятся все более и более «транзитивными», приближаясь к порядку $R = (v_1, \dots, v_n)$.

Последний тип задач получается при объединении линейных порядков. Каждый избиратель равновероятно выбирает порядок на вершинах. Эти порядки объединяются по уже известной схеме как разность между числом избирателей, предпочитающих v_j , и числом избирателей, предпочитающих v_i . Видно, что нет существенного различия между задачами с различным числом избирателей (рис. 7). Задачи со 100 или 101 избирателями ненамного сложнее задач с 3 или 4 избирателями. Индексы транзитивности остаются довольно высокими, что позволяет решать задачи размерности до 55 вершин за приемлемое время.

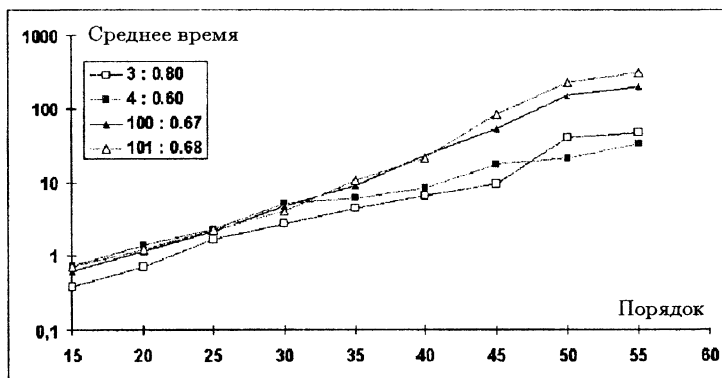


Рис. 7. Время решения задач, полученных объединением линейных порядков

Из полученных результатов следует, что максимальная размерность решаемых задач зависит от типа турнира. Для «очень трудных» турниров с индексами транзитивности около 0,1 для невзвешенных турниров

максимальная размерность менялась от 30 вершин до 40. На более легких задачах это значение достигает 100. Максимальная размерность для задачи Слейтера достигала 45.

Последнее замечание касается метода шума, который применяется перед методом ветвей и границ. На 5790 турнирах размерности до 100 вершин метод шума нашел оптимальное решение во всех случаях, кроме шести. Один турнир соответствовал задаче $P(0, 5; 1)$ и пять турниров — задачам, полученным объединением линейных порядков. Для решения этих задач методу шума потребовалось несколько секунд. Поскольку это стохастический метод, то при повторных запусках он, вероятно, нашел бы точное решение.

7. Выводы

Во многих практических задачах турниры имеют высокий индекс транзитивности, что позволяет решать задачи довольно большой размерности. Однако, когда число вершин больше 100, трудно решить задачу точно, поскольку необходимый объем памяти для множителей Лагранжа и требуемое вычислительное время становятся неприемлемо большими. В этом случае метод шума, как правило, дает хорошее приближенное решение. Комбинация метода шума с другими подходами, такими как лагранжева релаксация, метод ветвей и границ и т. д., для многих турниров приводит к хорошим результатам. Разработанное программное обеспечение доступно по адресу

<http://www.enst.fr/~charon/tournament/median.html>.

ЛИТЕРАТУРА

1. Aarts E. H. L., Lenstra J. K. (eds.) Local search in combinatorial optimization. Chichester: John Wiley & Sons, 1997.
2. Arditti D. Un nouvel algorithme de recherche d'un ordre induit par des comparaisons par paires // Data analysis and informatics. III. Amsterdam: North Holland, 1984. P. 323–343.
3. Barthélemy J.-P., Guénoche A., Hudry O. Median linear orders: heuristics and a branch and bound algorithm // European J. Oper. Res. 1989. V. 41, N 3. P. 313–325.
4. Barthélemy J.-P., Monjardet B. The median procedure in cluster analysis and social choice theory // Math. Social Sci. 1981. V. 1, N 3. P. 235–267.
5. Bermond J.-C. Ordres à distance minimum d'un tournoi et graphes partiels sans circuits maximaux // Math. Sci. Humaines. 1972. V. 37. P. 5–25.
6. Boenchenndorf K. Reihenfolgeprobleme. Königstein: Verl. Anton Hain, 1982. (Mathematical Systems in Economics; V. 74).

7. **Charon I., Germa A., Hudry O.** Encadrement de l'indice de Slater d'un tournoi à l'aide de ses scores // *Math., Inform. Sci. Humaines*. 1992. V. 118. P. 53–68.
8. **Charon I., Guénoche A., Hudry O., Woirgard F.** New results on the computation of median orders // *Discrete Math.* 1997. V. 165–166. P. 139–153.
9. **Charon I., Hudry O.** The noising method: a new method for combinatorial optimization // *Oper. Res. Lett.* 1993. V. 14, N 3. P. 133–137.
10. **Charon I., Hudry O.** Lamarckian genetic algorithms applied to the aggregation of preferences // *Ann. Oper. Res.* 1998. V. 80. P. 281–297.
11. **Charon I., Hudry O.** The noising methods: a generalization of some meta-heuristics. (Submitted for publication.)
12. **Charon I., Hudry O.** Automatic tuning of the noising methods. (Submitted for publication.)
13. **Christof T., Reinelt G.** Combinatorial optimization and small polytopes // *Topology*. 1996. V. 4, N 1. P. 1–64.
14. **Dell'Amico M., Maffioli F., Martello S. (eds.)** Annotated bibliographies in combinatorial optimization. Chichester: John Wiley & Sons, 1997.
15. **Grötschel M., Jünger M., Reinelt G.** A cutting plane algorithm for the linear ordering problem // *Oper. Res.* 1984. V. 32, N 6. P. 1195–1220.
16. **Guénoche A.** How to choose according to partial evaluations // *Advances in intelligent computing, IPMU'94*. Berlin: Springer-Verl., 1995. P. 611–618. (Lecture Notes in Comput. Sci.; V. 945).
17. **Hudry O.** Recherche d'ordres médians: complexité, algorithmique et problèmes combinatoires: PhD thesis. Paris: ENST, 1989.
18. **Kaas R.** A branch and bound algorithm for the acyclic subgraph problem // *European J. Oper. Res.* 1981. V. 8, N 4. P. 355–362.
19. **Karp R. M.** Reducibility among combinatorial problems // *Complexity of Computer Computations*. New York: Plenum Press, 1972. P. 85–103.
20. **Kemeny J. G.** Mathematics without numbers // *Daedalus*. 1959. V. 88. P. 577–591.
21. **Kendall M. G., Babington Smith B.** On the method of paired comparisons // *Biometrika*. 1940. V. 33. P. 239–251.
22. **Korte B., Oberhofer W.** Zwei Algorithmen zur Lösung eines komplexen Reihenfolgeproblems // *Unternehmensforschung*. 1968. V. 12. P. 217–231.
23. **Korte B., Oberhofer W.** Zur Triangulation von Input-Output Matrizen // *Jahrbuch Nationalökon. Statist.* 1969. V. 182. P. 398–433.
24. **Laporte G., Osman I. H.** Metaheuristics: a bibliography // *Ann. Oper. Res.* 1996. V. 63. P. 513–623.

25. **Lenstra H. W., Jr.** The acyclic subgraph problem // Technical report BW26. Amsterdam: Mathematisch Centrum, 1973.
26. **Marcotorchino J.-F., Michaud P.** Optimisation en analyse ordinale de données. Paris: Masson, 1979.
27. **Minoux M.** Mathematical programming: theory and algorithms. Chichester: John Wiley & Sons, Ltd., 1986.
28. **Moon J. W.** Topics on tournaments. New York: Holt; London: Rinehart and Winston, 1968.
29. **Reeves C. (ed.)** Modern heuristic techniques for combinatorial problems. London: McGraw-Hill, 1993.
30. **Remage R., Jr., Thompson W. A., Jr.** Maximum-likelihood paired comparison rankings // *Biometrika*. 1966. V. 53. P. 143–149.
31. **Slater P.** Inconsistencies in a schedule of paired comparisons // *Biometrika*. 1961. V. 48. P. 303–312.
32. **Wessels H.** Triangulation und Blocktriangulation von Input-Output Tabellen. Deutsches Institut für Wirtschaftsforschung: Beiträge zur Strukturforshung. Berlin: Heft 63, 1981.

Адрес авторов:

Charon I., Hudry O.
National Center
of Scientific Reseach ENST,
46, rue Barrault,
75634 Paris, cedex 13, France.
E-mail: hudry@infres.enst.fr

Статья поступила

26 июня 2000 г.,
переработанный вариант —
16 февраля 2001 г.