Эффективная точная модель для задачи дробного 0-1 программирования

Илья Бычков

Laboratory of Algorithms and Technologies for Network Analysis

Групповая технология

Обрабатывая деталей со схожими характеристиками (геометрия, производственный процесс, функции и прочее) вместе, мы можем снизить различные производственные издержки

Задача о разбиении производства на несколько кластеров, в каждом из которых обрабатывается своя группа деталей – cell formation problem

Cell Formation Problem

Классическая постановка задачи

Имеется производство, состоящее из **m** машин и **p** производимых деталей.
Процесс обработки задан матрицей **A**

$$a_{ij} = \begin{cases} 1, & \text{if machine } i \text{ processes part } j \\ 0, & \text{otherwise} \end{cases}$$

Задача: сгруппировать строки и столбцы матрицы **A** для получения так называемой блочно-диагональной структуры

Cell Formation Problem

Пример

	1	2	3	4	5	6	7
1	1	0	0	0	1	1	1
2	0	1	1	1	1	0	0
3	0	0	1	1	1	1	0
4	1	1	1	1	0	0	0
5	0	1	0	1	1	1	0

King & Nakornchai 5 x 7 instance

King & Nakornchai 5 x 7 solution

Cell Formation Problem

Актуальность

- Многие тестовые задачи до сих пор имеют лишь эвристические решения

- Cell formation problem идентична задаче о бикластеризации (biclustering), которая в свою очередь имеет широкое применение, например, для анализа данных в биологии (гены).

- Таким образом, алгоритмы и методы для cell formation problem могут быть использованы для различных задач бикластеризации

Cell Formation Problem

Целевые функции

1. Grouping Efficiency (Chandrasekharan & Rajagopalan, 1989)

$$\eta = q\eta_1 + (1-q)\eta_2,$$

Где

$$\eta_1 = \frac{n_1 - n_1^{out}}{n_1 - n_1^{out} + n_0^{in}} = \frac{n_1^{in}}{n^{in}},$$

$$\eta_2 = \frac{mp - n_1 - n_0^{in}}{mp - n_1 - n_0^{in} + n_1^{out}} = \frac{n_0^{out}}{n^{out}},$$

Cell Formation Problem

Целевые функции

2. Grouping Efficacy (Chandrasekharan & Rajagopalan, 1990)

$$\tau = \frac{n_1 - n_1^{out}}{n_1 + n_0^{in}} = \frac{n_1^{in}}{n_1 + n_0^{in}}.$$

Cell Formation Problem

Целевые функции

3. E + V (exceptional elements + voids)

Наименее популярная из приведенных мера, представляет собой сумму единиц вне диагонального блока + сумму нулей внутри диагонального блока

Cell Formation Problem

Пример

	1	2	3	4	5
1	1	0	0	1	0
2	0	1	1	1	0
3	0	0	1	0	0
4	0	0	0	1	1
5	0	1	0	1	1

Grouping Efficiency = 88.88%
Grouping Efficacy = 63.63%
E + V = 4

Cell Formation Problem

Ограничения

1. **Каждая машина может находиться ровно в одной ячейке, каждая часть должна обрабатываться ровно в одной ячейке**
2. **В подавляющем большинстве исследований запрещаются ячейки содержащие только машины или только детали (подобные решения сложно интерпретировать и использовать, хотя они могут дать некоторый выигрыш в целевой функции)**
3. **Ячейки с одной машиной и/или одной деталью как правило разрешены.**
4. **Число кластеров в классической постановке не задано, хотя существуют и иные формулировки.**

Cell Formation Problem

Формальное описание

Входные данные:

- m - число машин
- p - число деталей
- c - максимально возможное число ячеек (исходя из отсутствия пустых ячеек - min(m,p))

- матрица A, такая что:

$$a_{ij} = \begin{cases} 1, & \text{if machine } i \text{ processes part } j \\ 0, & \text{otherwise} \end{cases}$$

Cell Formation Problem

Формальное описание

Переменные (decision variables)

$$x_{ik} = \begin{cases} 1, & \text{if machine } i \text{ belongs to cell } k \\ 0, & \text{otherwise} \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{if part } j \text{ belongs to cell } k \\ 0, & \text{otherwise} \end{cases}$$

Cell Formation Problem

Формальное описание

Ограничения(constraints):

$$\sum_{k=1}^{c} x_{ik} = 1 \quad i = 1, \ldots, m$$

$$\sum_{k=1}^{c} y_{jk} = 1 \quad j = 1, \ldots, p$$

$$\sum_{i=1}^{m} x_{ik} \leq M \cdot \sum_{j=1}^{p} y_{jk} \quad k = 1, \ldots, c$$

$$\sum_{j=1}^{p} y_{jk} \leq M \cdot \sum_{i=1}^{m} x_{ik} \quad k = 1, \ldots, c$$

Cell Formation Problem

Формальное описание

Целевая функция(objective):

$$n_1^{in} = \sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{c} a_{ij}x_{ik}y_{jk} \qquad n_0^{in} = \sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{c}(1-a_{ij})x_{ik}y_{jk} \qquad n_1 = \sum_{i=1}^{m}\sum_{j=1}^{p} a_{ij}$$

$$max \quad \frac{n_1^{in}}{n_1 + n_0^{in}}$$

Cell Formation Problem

Точные подходы:

Рассмотрим 5 существующих точных методы решения cell formation problem.
Среди них 4 модели целочисленного линейного программирования и одна реализация метода ветвей и границ.

1. **MINpCUT model (Krushinsky and Goldengorin,2012)**
2. **MILP model by Elbenani and Ferland (2012)**
3. **MILP model by Brusco (2015)**
4. **Branch-and-Bound by Brusco (2015)**
5. **MILP model by Bychkov et. al (2014)**

Cell Formation Problem

Точные подходы:

MINpCUT model (Krushinsky and Goldengorin,2012)

Модель основана на графовом представлении и дальнейшем решении задачи о k-разбиении графа (то есть нахождении множества ребер, удаление которых приведет к распаду графа на k компонент связности).

В качестве целевой функции используется минимизация числа exceptional elements.

К сожалению, она не учитывает загрузку машин внутри кластеров.

Cell Formation Problem

Точные подходы:

MILP model by Elbenani and Ferland(2012)

Представили модель линейного программирования, применили алгоритм Динькельбаха для линеаризации целевой фукции (использована grouping efficacy).
Задача решается для фиксированного заранее числа производственных ячеек, что не позволяет назвать найденные решения глобальными оптимумами.
В целом, модель решается CPLEX достаточно быстро, тем не менее в некоторых случаях алгоритм был остановлен из-за ограничений по времени/памяти.

Cell Formation Problem

Точные подходы:

MILP model by Brusco (2015)

Представил модель основанную на two-mode clustering с некоторыми допущениями (одинаковое число кластеров по строкам и столбцам).

Целевая функция также линеаризована при помощи алгоритма Динкельбаха. Как и в модели Elbenani & Ferland(2012) используется фиксированное число ячеек, однако в данном случае авторы перебирают его для небольших примеров.

К сожалению, данная модель встречает затруднения даже на тестовых данных среднего размера.

Cell Formation Problem

Точные подходы:

Branch-and-Bound by Brusco (2015)

Второй подход описанный в предыдущей рассмотренной статье - реализация метода ветвей и границ для cell formation problem. Использует собственную эвристику (ILS) для получения начального решения. Данный подход способен решить примерно в 2 раза больше задач чем предыдущий. В целом, работает достаточно быстро на тестовых данных, где значение целевой функции 0.65-0.7.

Число ячеек также фиксированно, но рассмотрены интервалы для маленьких тестовых задач.

Кроме того, оба похода Бруско допускают существование пустых ячеек (без машин или деталей).

Cell Formation Problem

Точные подходы:

MILP model by Bychkov et. al (2014)

Наиболее похожа на поставленную в текущем исследовании задачу.

Число ячеек не фиксировано заранее.

Работает быстро на маленьких тестовых примерах, однако число переменных и ограничений в ней растет достаточно быстро - O(m^2n). Это не позволяет решить даже тестовые примеры средних размеров.

.

2 index MILP

2 index MILP

Входные данные:

- m - число машин
- p - число деталей
- матрица A, такая что:

$$a_{ij} = \begin{cases} 1, & \text{if machine } i \text{ processes part } j \\ 0, & \text{otherwise} \end{cases}$$

2 index MILP

2 index MILP

Переменные(decision variables):

$$x_{ij} = \begin{cases} 1, & \text{if machine } i \text{ and machine } j \text{ belongs to same cell} \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ik} = \begin{cases} 1, & \text{if machine } i \text{ and part } k \text{ belongs to the same cell} \\ 0, & \text{otherwise} \end{cases}$$

2 index MILP

2 index MILP

Ограничения(constraints):

$$2x_{ij} - z_{ik} - z_{jk} \geq -1 \quad i = 1, ..., m \quad j = 1, ..., m \quad k = 1, ..., p \tag{6}$$

$$z_{ik} - z_{jk} - x_{ij} \geq -1 \quad i = 1, ..., m \quad j = 1, ..., m \quad k = 1, ..., p \tag{7}$$

$$z_{jk} - z_{ik} - x_{ij} \geq -1 \quad i = 1, ..., m \quad j = 1, ..., m \quad k = 1, ..., p \tag{8}$$

$$\sum_{k=1}^{p} z_{ik} \geq 1 \quad i = 1, ..., m \tag{9}$$

$$\sum_{i=1}^{m} z_{ik} \geq 1 \quad k = 1, ..., p \tag{10}$$

Objective linearization

Метод 1.

Вводим новую переменную $$y = \frac{1}{n_1 + n_0^{in}}$$

Чтобы обеспечить это значение введем неравенство:

$$n_1 \cdot y + n_0^{in} \cdot y = 1$$

Objective linearization

Метод 1.

$$n_1 \cdot y + \sum_{i=1}^{m} \sum_{k=1}^{p} (1 - a_{ik}) \cdot z_{ik} \cdot y = 1$$

Так как n1 - константа, нам остается линеаризовать произведение Z_ik * y, что мы делаем добавлением переменной W_ik и добавлением 3х неравенств:

$$w_{ik} \leq z_{ik}$$

$$w_{ik} \leq y$$

$$w_{ik} \geq z_{ik} + y - 1$$

$$w_{ik} = \begin{cases} y, & \text{if } z_{ik} = 1 \\ 0, & \text{otherwise} \end{cases}$$

Objective linearization

Метод 1.

Целевая функция

$$max \sum_{i=1}^{m} \sum_{k=1}^{p} a_{ik} w_{ik}$$

Objective linearization

Метод 2.

Grouping Efficacy

$$\tau = \frac{n_1 - n_1^{out}}{n_1 + n_0^{in}} = \frac{n_1^{in}}{n_1 + n_0^{in}}.$$

$$0 \leq n_0^{current} \leq \left\lfloor \frac{1-\tau}{\tau} n_1 \right\rfloor$$

where τ - efficacy value of the best solution known for the particular problem instance and n_1 - total number of operations in this instance.

Objective linearization

Метод 2.

Целевая функция

$$max \quad \sum_{i=1}^{m} \sum_{k=1}^{p} a_{ik} z_{ik}$$

Дополнительное ограничение на число нулей:

$$\sum_{i=1}^{m} \sum_{k=1}^{p} (1 - a_{ik}) z_{ik} = n_0^{current}$$

Objective linearization

Сравнение 2 index MILP с разными функциями линеаризации

#	m	p	Time, sec		Times faster
			Method 1	Fixing zeroes	
1	5	7	0.0223	0.0056	3.9821
2	5	7	0.128	0.0336	3.8095
3	5	18	0.9871	0.0824	11.9794
4	6	8	0.088	0.0322	2.7329
5	7	11	0.7835	0.1989	3.9392
6	7	11	0.3531	0.0569	6.2056
7	8	12	0.6975	0.1659	4.2043
8	8	20	0.2598	0.1492	1.7413
9	8	20	242.86	16.5	14.7188
10	10	10	0.3462	0.094	3.6830

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Далее посмотрим на результаты сравнения 4х подходов описанных ранее с предложенной 2 index MILP моделью и линеаризацией фиксированием знаменателя целевой функции:

1. **MILP by Elbenani & Ferland(2012)**
2. **MILP by Brusco (2015)**
3. **MILP by Bychkov et. Al (2014)**
4. **BnB by Brusco (2015)**

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Для вычислительных экпериментов использовалось 2 набора тестовых данных - Testset A и Testset B.

Testset A - набор из 35 задач размерами от 5x7 до 40x100. Этот набор популярен в литературе и используется многими исследователями.

Testset B - более редкие тестовые задачи, этот набор был составлен мной, содержит 46 тестовых задач из литературы, размерами от 4x5 до 50x150.

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Классификация задач:

- small-sized (less than 10 machines)
- medium-sized (from 10 to 20 machines)
- large-sized (20 machines or greater)

Table 1: Testets

	small	medium	large
Testset A	9	8	18
Testset B	18	19	9

Computational results

Table 6: Testset A - Computational results

#	Time, sec					Efficacy				
	Elbenani & Ferland	3 index	Brusco MILP	2 index	Brusco BnB	Elbenani & Ferland (cells)	Bychkov et.	Brusco MILP (cells)	2 index	Brusco BnB (cells)
A1	2.3	0.63	0.21	0.0056	< 0.03	0.8235(2)	0.8235	0.8235 (2,3,4)	0.8235	0.8235(2,3,4)
A2	1.6	2.29	0.49	0.0336	< 0.03	0.6957(2)	0.6957	0.6957(2,3,4)	0.6957	0.6957(2,3,4)
A3	3.1	5.69	residual	0.0824	residual	0.7959(2)	0.7959	residual	0.7959	residual
A4	2	1.86	residual	0.0322	residual	0.7692(2)	0.7692	residual	0.7692	residual
A5	30.6	9.14	15.79	0.1989	0.6	0.6087(5)	0.6087	0.6087(2,3,4,5,6)	0.6087	0.6087(2,3,4,5,6)
A6	4.3	5.15	2.69	0.0569	< 0.06	0.7083(4)	0.7083	0.7083(2,3,4,5)	0.7083	0.7083(2,3,4,5)
A7	9.6	13.37	6.46	0.1659	0.08	0.6944(4)	0.6944	0.6944(2,3,4,5)	0.6944	0.6944(2,3,4,5)
A8	3.1	18.33	1.69	0.1492	< 0.03	0.8525(3)	0.8525	0.8525(2,3,4)	0.8525	0.8525(2,3,4)
A9	3.5	208.36	42.44	16.5	35.86	0.5872(2)	0.5872	0.5872(2,3,4)	0.5872	0.5872(2,3,4)
A10	1.1	6.25	33.89	0.0974	< 0.08	0.75(5)	0.75	0.75(2,3,4,5,6)	0.75	0.75(2,3,4,5,6)
A11	1.63	2.93	1.71	0.056	< 0.03	0.92(3)	0.92	0.92(2,3,4)	0.92	0.92(2,3,4)
A12	2188.7	259.19	residual	1.5025	residual	0.7206(7)	0.7206	residual	0.7206	residual
A13	593.2	179.21	residual	1.7443	residual	0.7183(7)	0.7183	residual	0.7183	residual
A14	15130.5	*	-	358.4992	residual	0.5326(8)	*	-	0.5326	residual
A15	252.5	*	-	14.2355	residual	0.6953(6)	*	-	0.6899	residual
A16	183232.5	*	-	283.6951	-	0.5753(8)	*	-	0.5753	-
A17	2345.6	*	-	149.5414	20840.55	0.5773(9)	*	-	0.5773	0.5773(9)
A18	*	*	-	*	-	0.4306(5)	*	-	*	-
A19	131357.5	*	-	10490.87	1375608.66	0.5081(7)	*	-	0.5081	0.5081(7)
A20	31.1	*	-	13.8466	residual	0.7791(5)	*	-	0.7791	residual
A21	14583.6	*	-	1400.55	-	0.5798(5)	*	-	0.5798	-
A22	11.3	1.64	-	0.0884	< 0.01	1(7)	1	-	1	1(7)
A23	230.7	*	-	7.1063	42.29	0.8511(7)	*	-	0.8511	0.8511(7)
A24	1101.1	*	-	49.99	208158.02	0.7351(7)	*	-	0.7351	0.7351(7)
A25	*	*	-	8883.98	-	0.5329(11)	*	-	0.5329	-
A26	*	*	-	*	-	0.4828(12)	*	-	*	-
A27	*	*	-	*	-	0.457(12)	*	-	*	-
A28	958714.1	*	-	47811.75	-	0.5482(5)	*	-	0.5482	-
A29	*	*	-	*	-	0.4675(10)	*	-	*	-
A30	378300	*	-	398.42	-	0.6331(14)	*	-	0.6331	-
A31	*	*	-	*	-	0.6012(13)	*	-	*	-
A32	*	*	-	*	-	0.5055(14)	*	-	*	-
A33	*	*	-	*	-	0.4467(17)	*	-	*	-
A34	268007.6	*	-	*	-	0.6064(3)	*	-	*	-
A35	7365.3	*	-	1072.397	-	0.8403(10)	*	-	0.8403	-

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Testset A

2 index MILP vs Elbenani & Ferland MILP(2012)

Оба алгоритма решили по 27 задач из 35.
Несмотря на то, что моя модель решает задачу для всех возможных количеств кластеров в 26 из 27 случаев она работает быстрее. Разница во времени вычислений достигает 1456 раз.
В тестовом примере A15 решение Elbenani & Ferland лучше, чем мое, что может быть объяснено использованием им некорректных входных данных (мои данные были сверены с оригиналом).
По 8 задач не решили оба подхода.

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Testset A

2 index MILP vs Bychkov et. Al MILP(2014)

Моя предыдущая модель была способна решить лишь 14 из 35 самых маленьких по размеру тестовых задач (одна из задач 24x40, но с целевой функцией 1).
За счет значительного сокращения числа переменных и ограничений удалось получить выигрыш во времени вычислений на всех 14 примерах от 12.62 до 172 раз.

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Testset A

2 index MILP vs Brusco MILP(2015)

В своей статье Brusco приводит вычислительные эксперименты для 21 из 35 тестовых примеров сета A.
При этом предложенная им модель линейного программирования решила лишь 13 самых маленьких примера из 21.
При этом в 4х случаях решения содержат пустые клетки, так что я сравнивал время выполнения лишь для оставшихся 9ти.
Во всех этих 9ти случая 2 index MILP быстрее до 347.94 раз.

Computational results

Сравнение 2 index MILP + linearization с другими подходами

Testset A

2 index MILP vs Brusco BnB(2015)

Реализация метода ветвей и границ работает быстро для большинства инстансов маленького размера (6) с целевой функцией > 0.65. На этих примерах мы проигрываем с разницей до 4.97 раз.
На остальных 8 примерах среднего размера с более сложной структурой моя модель работает быстрее с разницей до 4163.99 раз.

Computational results

Table 7: Testset B - Computational results

#	Time			Efficacy			
	Brusco MILP	Brusco BnB	2 index	Heuristic bound	Brusco MILP (cells)	Brusco BnB (cells)	2 index
B1	-	-	0.0023	0.9	-	-	0.9
B2	-	-	0.0026	0.9	-	-	0.9
B3	-	-	0.0021	0.9	-	-	0.9
B4	-	-	0.0013	1	-	-	1
B5	-	-	0.0044	0.8182	-	-	0.8182
B6	-	-	0.0212	0.7059	-	-	0.7059
B7	-	-	0.0184	0.8095	-	-	0.8095
B8	-	-	0.0282	0.7222	-	-	0.7222
B9	-	-	0.1845	0.6071	-	-	0.6071
B10	-	-	0.004	0.8889	-	-	0.8889
B11	-	-	0.0138	0.75	-	-	0.75
B12	-	-	0.0427	0.7391	-	-	0.7391
B13	2.23	<0.04	0.1121	0.7037	0.6857(2,3,4,5)	0.6857(2,3,4,5)	0.7037
B14	-	-	0.0357	0.8148	-	-	0.8148
B15	-	-	0.0479	0.7222	-	-	0.7222
B16	-	-	0.0929	0.7576	-	-	0.7576
B17	-	-	0.0222	0.9	-	-	0.9
B18	-	-	0.1766	0.7273	-	-	0.7273
B19	-	-	0.0182	0.85	-	-	0.85
B20	-	-	0.0512	0.8276	-	-	0.8276
B21	-	-	2.1397	0.5962	-	-	0.5962
B22	-	-	0.0028	1	-	-	1
B23	-	-	5.9841	0.6404	-	-	0.6404
B24	-	-	0.049	0.8667	-	-	0.8667
B25	3400.46	129.53	3.9625	0.7391	0.7(2,3,4,5,6,7,8)	0.7(2,3,4,5,6,7,8)	0.7391
B26	residual	residual	0.2202	0.7333	residual	residual	0.7333
B27	-	-	4.4872	0.6552	-	-	0.6552
B28	residual	residual	43.3758	0.5765	residual	residual	0.5765
B29	residual	residual	71.306	0.5666	residual	residual	0.5699
B30	-	-	6.959	0.623	-	-	0.623
B31	residual	residual	0.7742	0.8	residual	residual	0.8
B32	-	-	0.004	1	-	-	1
B33	21.04	<0.28	0.5226	0.871	0.871(2,3,4,5)	0.871(2,3,4,5)	0.871
B34	17.22	0.64	0.3031	0.8333	0.8182(2,3,4,5)	0.8182(2,3,4,5)	0.8333
B35	510.19	4.66	3.0872	0.7258	0.7258(2,3,4,5)	0.7258(2,3,4,5)	0.7258
B36	-	-	1.3666	0.8111	-	-	0.8111
B37	-	-	256.0387	0.5619	-	-	0.5673
B38	-	-	0.8994	0.76	-	-	0.76
B39	-	-	3.1421	0.7528	-	-	0.7528
B40	-	-	1374.6042	0.6068	-	-	0.6068
B41	-	-	33.7258	0.7347	-	-	0.7347
B42	-	-	280.6754	0.6729	-	-	0.6729
B43	-	-	668.6616	0.573	-	-	0.573
B44	-	-	95.9122	0.7358	-	-	0.7358
B45	-	-	31144.1074	0.6798	-	-	0.6798
B46	-	-	*	0.6193	-	-	*

Testset B - Results

Общих тестовых данных всего 9 из 46 примеров.

В 4х из них появляются пустые ячейки - их я не брал в сравнение.

В 2х из оставшихся пяти результаты целевой функции совпадают, в остальных 3х наше решение лучше чем результаты как MILP модели Brusco, так и BnB.

Во всех 5ти случаях моя модель быстрее до 858.1 раз чем Brusco MILP

Если говорить о BnB, в 2х случаях моя модель проигрывает, в 3х выигрывает.

Future work

Future work

- попробовать использовать подход с построением оценки на знаменатель и перебором его фиксированных значений исходя из оценки лучшего решения

- применить 2 index MILP для решения различных задач бикластеризации

- рассмотреть формулировки с реальными ограничениями

Спасибо за внимание!