Greedy algorithms for computing the Birkhoff-von Neumann decomposition

Damien Lesens
Supervised by Bora Uçar and Jérémy E. Cohen

ROMA team, LIP, ENS de Lyon

June 2024

Plan

1. Background
   - Birkhoff theorem
   - Applications
   - Heuristics
   - Sparse coding
2. GompBvN
   - Algorithm
   - Results
3. Generalisation of the BvN decomposition
4. Conclusion

Birkhoff theorem

Definition

An $n \times n$ matrix A is **doubly stochastic** if $a_{ij} \geq 0$, row sums and column sums are 1

Theorem (Birkhoff,von Neumann)

For a doubly stochastic matrix $\mathbf{A}$*, there exist* $\alpha_1, \alpha_2, \ldots, \alpha_k \in (0, 1]$ *with* $\sum_{i=1}^{k} \alpha_i = 1$ *and* $n \times n$ *permutation matrices* $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_k$ *such that*

$$\mathbf{A} = \alpha_1 \mathbf{P}_1 + \alpha_2 \mathbf{P}_2 + \cdots + \alpha_k \mathbf{P}_k$$

Applications

- Generalises to a large class of matrix $\rightarrow$ numerical applications
- Routing traffic in data centers (circuit switches)
- Assignment problems and economics

Applications

- Generalises to a large class of matrix $\rightarrow$ numerical applications
- Routing traffic in data centers (circuit switches)
- Assignment problems and economics

All these applications gain in efficiency
if the number of components k is small

The BvN decomposition is not unique

The problem

Sparse BvN decomposition problem

Given a doubly stochastic matrix $\mathbf{A}$

find a Birkhoff-von Neumann decomposition

$$\mathbf{A} = \alpha_1 \mathbf{P}_1 + \alpha_2 \mathbf{P}_2 + \cdots + \alpha_k \mathbf{P}_k$$

such that k is minimum

The problem

Sparse BvN decomposition problem

Given a doubly stochastic matrix $\mathbf{A}$

find a Birkhoff-von Neumann decomposition

$$\mathbf{A} = \alpha_1 \mathbf{P}_1 + \alpha_2 \mathbf{P}_2 + \cdots + \alpha_k \mathbf{P}_k$$

such that k is minimum

- Dufossé and Uçar proved that the problem is NP-complete
- Design heuristics

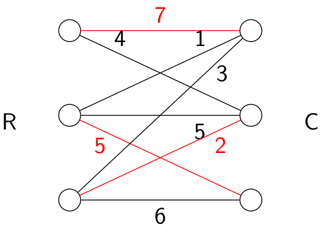
Known heuristics

- represent a doubly stochastic matrix as a bipartite graph
- Algorithm: *Birkhoff heuristic*
  - find a perfect matching in this graph
  - the coefficient is the minimum entry in the permutation
  - update and continue until all entries are zero

$$R \overset{C}{\begin{pmatrix} 7 & 4 & 0 \\ 1 & 5 & 5 \\ 3 & 2 & 6 \end{pmatrix}}$$

$$= 2\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 5 & 4 & 0 \\ 1 & 5 & 3 \\ 3 & 0 & 6 \end{pmatrix}$$

(a) Matrix representation



(b) Graph representation

Hard cases

- heuristics differ by the way the matching is chosen at each step
  $\rightarrow$ Dufossé and Uçar 2016
- inherent limitation : set an entry to zero at each step

Hard cases

- heuristics differ by the way the matching is chosen at each step
  $\rightarrow$ Dufossé and Uçar 2016
- inherent limitation : set an entry to zero at each step
- hard instance:

a	b	c	d	e	f	g	h	i	j
1	2	4	8	16	32	64	128	256	512

$$\mathbf{A} = \frac{1}{1023}\begin{pmatrix} a+b & d+i & c+h & e+j & f+g \\ e+g & a+c & b+i & d+f & h+j \\ f+j & e+h & d+g & b+c & a+i \\ d+h & b+f & a+j & g+i & c+e \\ c+i & g+j & e+f & a+h & b+d \end{pmatrix}$$

The optimal is 10, which will never be reached by any Birkhoff heuristic (Dufossé et al. 2018)

Sparse coding

- New family of heuristics which take inspiration from the field of *sparse coding*
- Sparse coding problem:

  Given an observation $\mathbf{a} \in \mathbb{R}^d$, linear combination of atoms coming from a dictionary $\mathbf{M} \in \mathbb{R}^{d \times k}$, find coefficients $\mathbf{x} \in \mathbb{R}^k$ such that $\mathbf{a} \approx \mathbf{Mx}$ and $\mathbf{x}$ is the sparsest, i.e., it has as few non-zero entries as possible

Sparse coding

BvN decomposition with min. terms as a sparse coding problem, introduced by Dufossé et al. 2018

The permutations are ordered arbitrarily as $\mathbf{P}_1$, $\mathbf{P}_2$,..., $\mathbf{P}_{n!}$

$$\mathbf{M} = (\text{vec}(\mathbf{P}_1) | \text{vec}(\mathbf{P}_2) | \cdots | \text{vec}(\mathbf{P}_{n!}))$$

Define $\mathbf{a} = \text{vec}(\mathbf{A})$ and solve the sparse coding problem

$$\mathbf{a} = \mathbf{M}\mathbf{x}$$

Algorithm: GompBvN

GompBvN: *Adaptation* of the Orthogonal Matching Pursuit (OMP) algorithm to the BvN decomposition.

After much *modifications* and *optimizations*:

```
Let i ← 1, S ← ∅, x ← 0
while has not converged do
    A^(i) ← A − Σ_{P_j ∈ S} x_j^(i−1) P_j
    find a perfect matching P_i ⊆ A^(i)          ▷ OMP_1
    S ← S ∪ {P_i}
    recompute coefficients x                      ▷ OMP_2
    i ← i + 1
end while
```

Algorithm: GompBvN

- By recomputing coefficients, we mean finding the "best" approximation given the permutations matrices already found. One can solve in OMP_2
  - $\min_{\mathbf{x}} \|\mathbf{A} - \sum_{\mathbf{P}_j \in S} x_j \mathbf{P}_j\|_2^2$: this gives a quadratic program
  - $\min_{\mathbf{x}} \|\mathbf{A} - \sum_{\mathbf{P}_j \in S} x_j \mathbf{P}_j\|_1$: this gives a linear program
- In OMP_1, we pick a matching: take the best option from literature on Birkhoff heuristics (e.g. bottleneck matching, Dufossé and Uçar 2016)
- If we compute x_i and do not optimize on $\mathbf{x}$ we get a Birkhoff heuristic

Results

- GompBvN performs similarly than the best Birkhoff-heuristic on matrices appearing in real-life applications
- Solves optimally instances that were previously out of reach:
  We have a class of matrix on which GompBvN will always be nearly 2 times better than Birkhoff

Size	Optimum	Birkhoff	GompBvN
100	10	19	11
200	15	29	16
500	20	39	21

Example of a computation

Both algorithm

$$\begin{pmatrix} 3 & 264 & 132 & 528 & 96 \\ 80 & 5 & 258 & 40 & 640 \\ 544 & 144 & 72 & 6 & 257 \\ 136 & 34 & 513 & 320 & 20 \\ 260 & 576 & 48 & 129 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 264 & 132 & 15 & 96 \\ 80 & 5 & 258 & 40 & 127 \\ 31 & 144 & 72 & 6 & 257 \\ 136 & 34 & 0 & 320 & 20 \\ 260 & 63 & 48 & 129 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 7 & 132 & 15 & 96 \\ 80 & 5 & 1 & 40 & 127 \\ 31 & 144 & 72 & 6 & 0 \\ 136 & 34 & 0 & 63 & 20 \\ 3 & 63 & 48 & 129 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 7 & 5 & 15 & 96 \\ 80 & 5 & 1 & 40 & 0 \\ 31 & 17 & 72 & 6 & 0 \\ 9 & 34 & 0 & 63 & 20 \\ 3 & 63 & 48 & 2 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 7 & 5 & 15 & 33 \\ 17 & 5 & 1 & 40 & 0 \\ 31 & 17 & 9 & 6 & 0 \\ 9 & 34 & 0 & 0 & 20 \\ 3 & 0 & 48 & 2 & 10 \end{pmatrix}$$

Birkhoff

$$\to \begin{pmatrix} 3 & 7 & 5 & 15 & 2 \\ 17 & 5 & 1 & 9 & 0 \\ 0 & 17 & 9 & 6 & 0 \\ 9 & 3 & 0 & 0 & 20 \\ 3 & 0 & 17 & 2 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 7 & 5 & 0 & 2 \\ 2 & 5 & 1 & 9 & 0 \\ 0 & 2 & 9 & 6 & 0 \\ 9 & 3 & 0 & 0 & 5 \\ 3 & 0 & 2 & 2 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 0 & 5 & 0 & 2 \\ 2 & 5 & 1 & 2 & 0 \\ 0 & 2 & 2 & 6 & 0 \\ 2 & 3 & 0 & 0 & 5 \\ 3 & 0 & 2 & 2 & 3 \end{pmatrix} \to \begin{pmatrix} 3 & 0 & 2 & 0 & 2 \\ 2 & 2 & 1 & 2 & 0 \\ 0 & 2 & 2 & 3 & 0 \\ 2 & 3 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 & 3 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 2 & 0 & 2 \\ 2 & 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 3 & 0 \\ 2 & 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

GompBvN

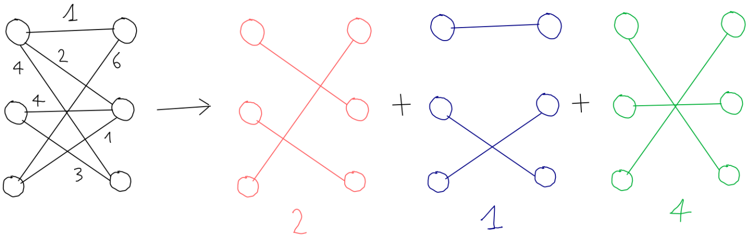
$$\to \begin{pmatrix} 3 & 7 & 3 & 17 & 0 \\ 17 & 5 & 1 & 7 & 0 \\ 0 & 15 & 9 & 6 & 0 \\ 7 & 1 & 2 & 0 & 20 \\ 3 & 2 & 15 & 0 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 7 & 4 & 0 & 1 \\ 1 & 5 & 1 & 8 & 0 \\ 0 & 0 & 9 & 6 & 0 \\ 8 & 2 & 1 & 0 & 4 \\ 3 & 1 & 0 & 1 & 10 \end{pmatrix} \to \begin{pmatrix} 3 & 0 & 4 & 0 & 0 \\ 0 & 5 & 2 & 0 & 0 \\ 0 & 0 & 0 & 6 & 1 \\ 0 & 2 & 1 & 0 & 4 \\ 4 & 0 & 0 & 1 & 2 \end{pmatrix} \to \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Publication

This work was accepted for publication at EUSIPCO 2024
"Orthogonal Matching Pursuit-based algorithm for the Birkhoff-von Neumann decomposition"

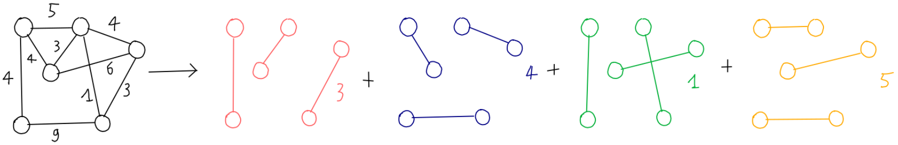
Generalisation of the BvN decomposition

Given **bipartite graph** and a weighting in the convex hull of its perfect matchings,
$\rightarrow$ find a decomposition



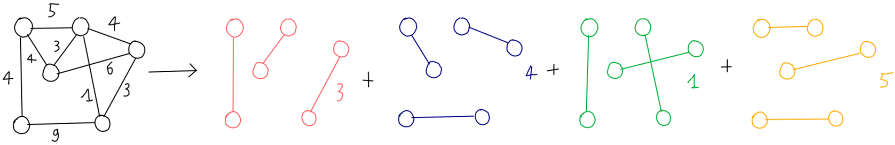
Generalisation of the BvN decomposition

Given **GENERAL graph** and a weighting in the convex hull of its perfect matchings,
$\rightarrow$ find a decomposition



Generalisation of the BvN decomposition

Given **GENERAL graph** and a weighting in the convex hull of its perfect matchings,
$\rightarrow$ find a decomposition



- cannot choose any perfect matching
- coefficient is not always the minimum edge value

Generalisation of the BvN decomposition

- V. Vazirani showed in 2020 that the problem is in $\mathcal{P}$
- Issues:
  - For $G = (V, E)$, $n = |V|$, $m = |E|$, it costs $O(n^2 m^3)$ max-flow min-cut computations
  - not implementable easily

Generalisation of the BvN decomposition

- Vazirani showed in 2020 that the problem is in $\mathcal{P}$
- Issues:
  - ~~$O(n^2 m^3)$ max-flow min-cut computations~~

    $\rightarrow O(n^3 \log(n) + n^2 m)$ max-flow min-cut
  - ~~not implementable easily~~

    $\rightarrow$ first implementation for the problem in Python

Conclusion

- New family of heuristics for the sparse BvN decomposition problem based on the sparse coding problem
  - New technique: recompute coefficients at each step
  - Strictly extends the state of the art for the problem
- First implementation for the generalised BvN decomposition problem
- Future work: extend both algorithms