Sparse AD

Benoît Legat

☐ Full Width Mode ☐ Present Mode

Table of Contents

- An Illustrated Guide to Automatic Sparse Differentiation
- Revisiting Sparse Matrix Coloring and Bicoloring

Sparse Jacobian

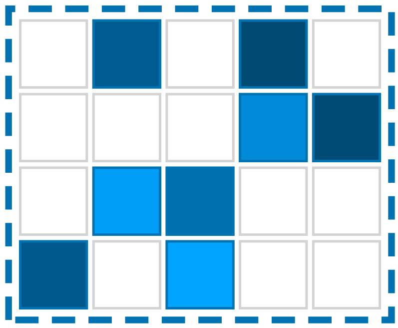
Suppose the Jacobian is **sparse**.

- The *sparsity pattern* (rows and columns of nonzeros) is **known**
- You want to determine the **value** of these nonzeros with AD

Consider the 4×5 sparse matrix on the right:

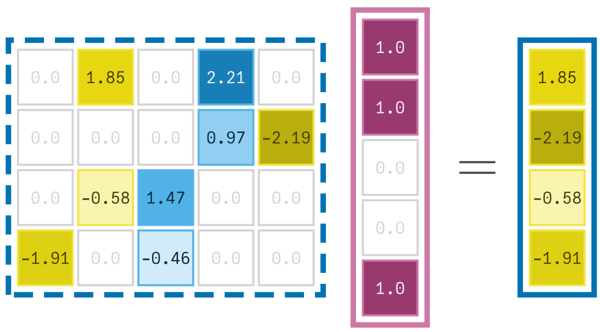
- Forward mode would require 5 JVP as there are 5 columns
- Reverse mode would require 4 VJP as there are 4 rows

Can we do better using the known sparsity pattern ?

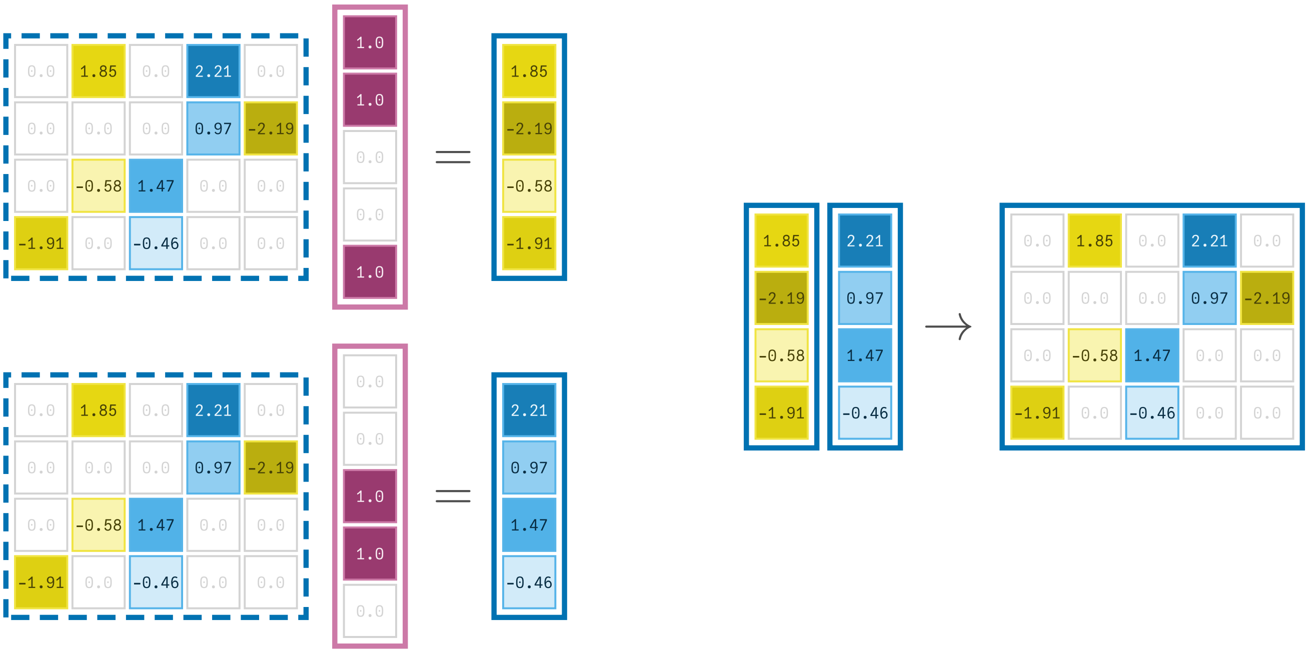


Three columns in just one JVP

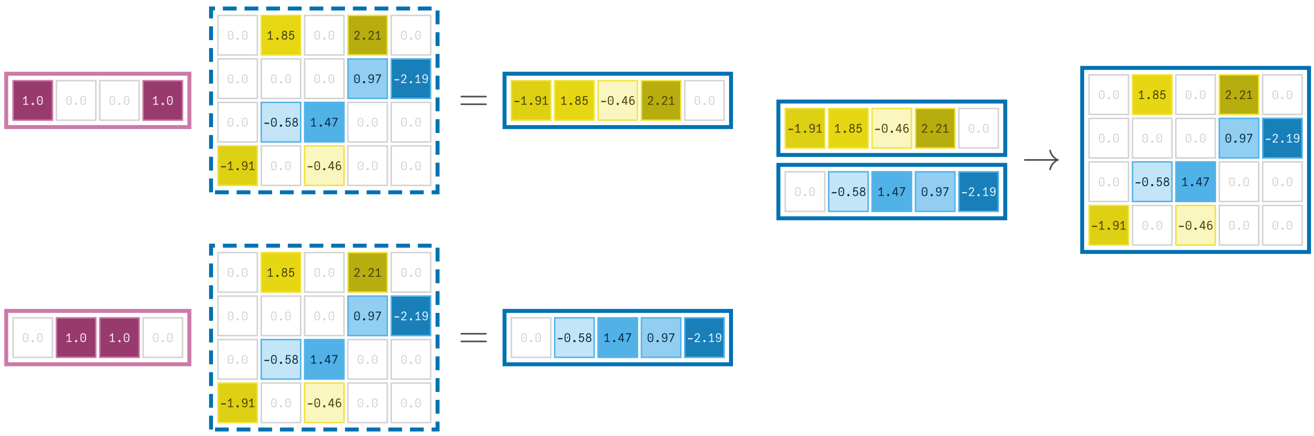
- The columns 1, 2 and 5 do not share any rows with nonzero entries.
- So their entries can be recovered unambiguously with just one JVP!



Whole Jacobian in just two JVP



The same applies to reverse mode



Sparsity detection of Jacobian

Simple operator overleading implementation

```
struct MyGradientTracer
    indexset::Set{Int}
end
```

```
Base.:+(a::MyGradientTracer, b::MyGradientTracer) =
MyGradientTracer(union(a.indexset, b.indexset))
```

```
Base.:*(a::MyGradientTracer, b::MyGradientTracer) =
MyGradientTracer(union(a.indexset, b.indexset))
```

```
Base.:/(a::MyGradientTracer, b::Number) = a
```

The sign function is constant over $x > 0$ and $x < 0$ so its output carries **no** variables

```
Base.sign(x::MyGradientTracer) = MyGradientTracer(Set()) # return empty index set
```

Scalar example

```
f (generic function with 1 method)
```

```
f(x) = x[1] + x[2]*x[3] * sign(x[4])
```

```
xtracer =
▸[MyGradientTracer(Set([1])), MyGradientTracer(Set([2])), MyGradientTracer(Set([3])), MyGradie
```

```
xtracer = [
    MyGradientTracer(Set(1)),
    MyGradientTracer(Set(2)),
    MyGradientTracer(Set(3)),
    MyGradientTracer(Set(4)),
]
```

```
ytracer = ▸MyGradientTracer(Set([2, 3, 1]))
```

```
ytracer = f(xtracer)
```

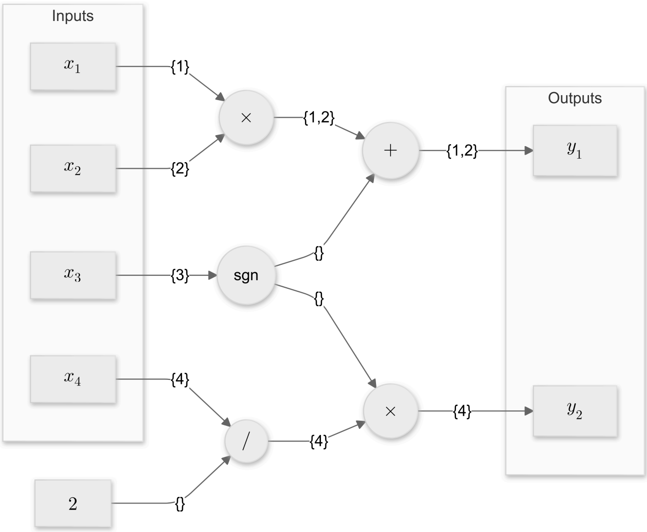
Sparsity detection with multiple outputs

```
F (generic function with 1 method)
```

```
F(x) = [x[1]*x[2]+sign(x[3]), sign(x[3]) * x[4]/2]
```

```
[MyGradientTracer(Set([2, 1])), MyGradientTracer(Set([4]))]
```

```
F(xtracer)
```

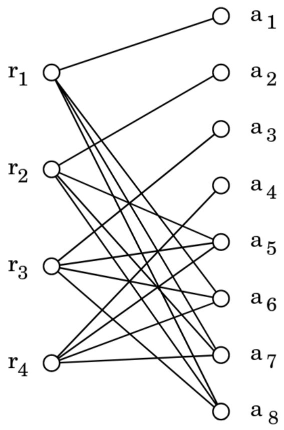


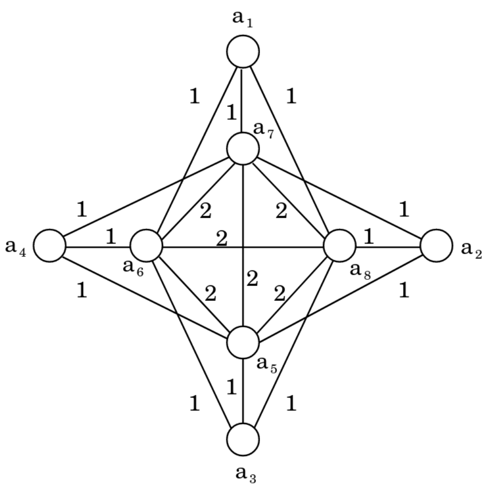
What color is your Jacobian ?

Given a sparse matrix A, two graphs represent its sparsity:

- Column intersection graph, used in Software for estimating sparse Jacobian matrices, 1984
- Bipartite graph, used in Colpack and SparseMatrixColoring

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 & a_{16} & a_{17} & a_{18} \\ 0 & a_{22} & 0 & 0 & a_{25} & 0 & a_{27} & a_{28} \\ 0 & 0 & a_{33} & 0 & a_{35} & a_{36} & 0 & a_{38} \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & a_{47} & 0 \end{bmatrix}$$





See Section 3.4 of What color is your Jacobian ?

Coloring problems

Given a graph $G(V, E)$,

- Nodes u, v are distance k neighbors if there exists a path from u to v of length at most k.
- A *distance-k* coloring : mapping $\phi : V \to \{1, \ldots, p\}$ such that $\phi(u) \neq \phi(v)$ whenever u, v are distance-k neighbors.
- *k-chromatic number* $\xi_k(G)$: mininum p such that $\exists$ distance-k coloring with p colors.
- Distance-k coloring problem : Find distance-k coloring with fewest colors.
- For every fixed integer $k \geq 1$, the distance-k graph coloring problem is NP-hard.

See Section 2.1, 2.2, 3.2 of What color is your Jacobian ?

Formulation as coloring problem

Theorem 3.5 A coloring of the columns is distance-2 in the bipartite graph iff columns of the same color are structurally orthogonal.

Lemma 3.7 The column intersection graph is the square of the biparted graph restricted to its column vertices.

Lemma 2.1 A coloring is distance-k in G iff it is distance-1 in G^k.

Example

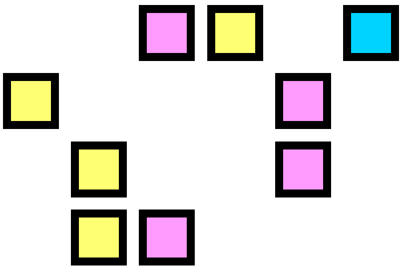
Consider the following sparsity pattern of the Jacobian matrix:

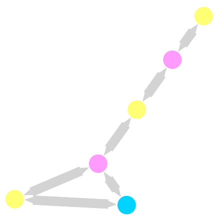
Taken from a tutorial of the SparseMatrixColorings' doc

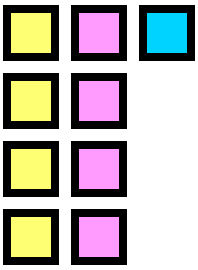
```
S = 4×6 SparseMatrixCSC{Int64, Int64} with 9 stored entries:
     ⋅  ⋅  1  1  ⋅  1
     1  ⋅  ⋅  ⋅  1  ⋅
     ⋅  1  ⋅  ⋅  1  ⋅
     ⋅  1  1  ⋅  ⋅  ⋅
```

Color's columns are structurally orthogonal

Jacobian from 3 JVP







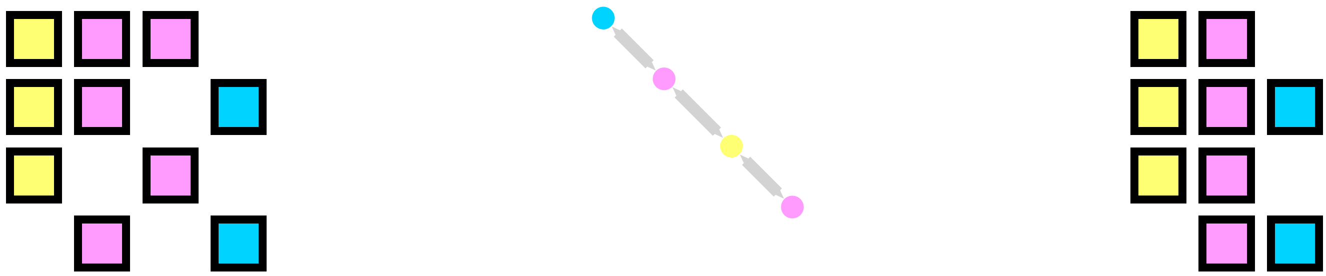
What color is your Hessian ?

How many HVP do we need to find the values of this sparse **symmetric** matrix ?

```
ijkl = 4×4 SparseMatrixCSC{Int64, Int64} with 10 stored entries:
        1  2  3  ⋅
        2  4  ⋅  5
        3  ⋅  6  ⋅
        ⋅  5  ⋅  7
```

Section 2.4 Consider the *adjacency graph* G of a matrix A be the graph whose adjacency matrix has same sparsity pattern as A. So i and j are adjacent iff a_{ij} is nonzero.

Need 3 colors for a path of 4 vertices



Star coloring

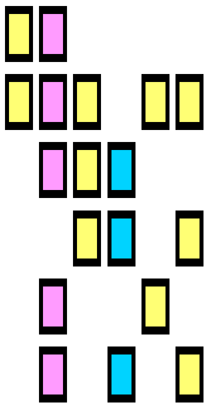
Definition 4.5 A mapping $\phi : V \to \{1, \ldots, p\}$ is a *star coloring* if

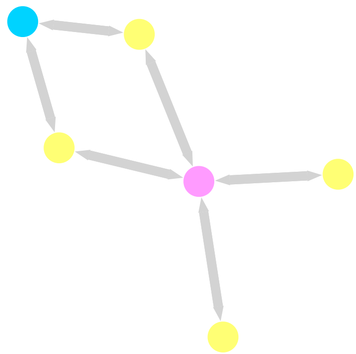
1. ϕ is a distance-1 coloring
2. every **path of 4 vertices** uses at least 3 colors

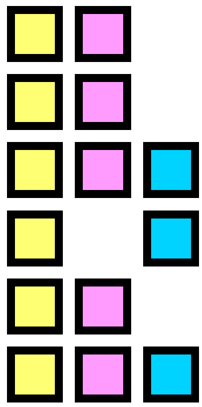
Theorem 4.6 Let A be a symmetric matrix with **nonzero diagonal elements**, G be its adjacency matrix. A mapping ϕ is a star coloring of the adjacency graph iff it induces a symmetrically structurally orthogonal partition of the columns of A.

Name star coloring comes from the fact that the subgraph induced by any pair of colors is a star.

Small Example



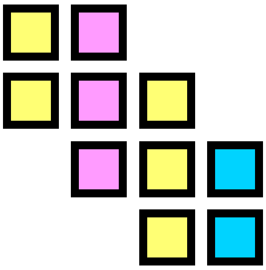


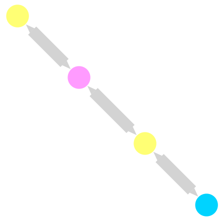


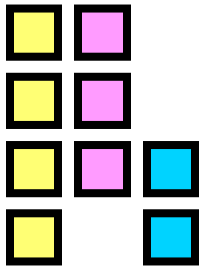
Less colors but nontrivial substitution

$$\begin{bmatrix} a_1 & a_2 & & \\ a_2 & a_3 & a_4 & \\ & a_4 & a_5 & a_6 \\ & & a_6 & a_7 \end{bmatrix}$$

With star coloring, 3 colors:



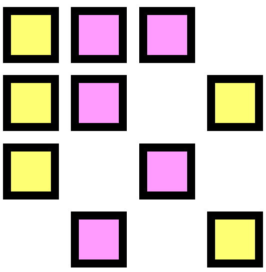


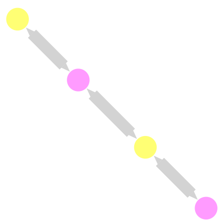


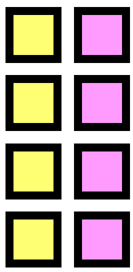
2 colors with substitutions

With 2 colors, our rwo forward tangents are

$$D^\top = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad AD = \begin{bmatrix} a_1 & a_2 \\ a_2 + a_4 & a_3 \\ a_5 & a_4 + a_6 \\ a_6 & a_7 \end{bmatrix}$$







Acyclic coloring

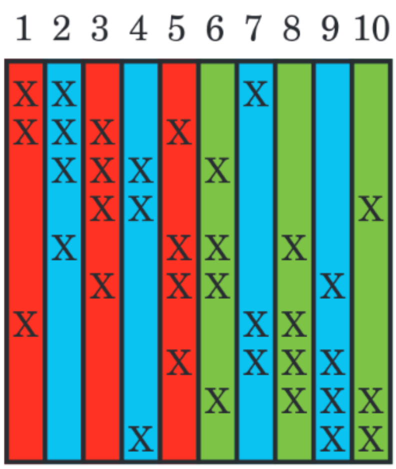
Definition 6.3 A mapping $\phi : V \to \{1, \dots, p\}$ is an *acyclic coloring* if

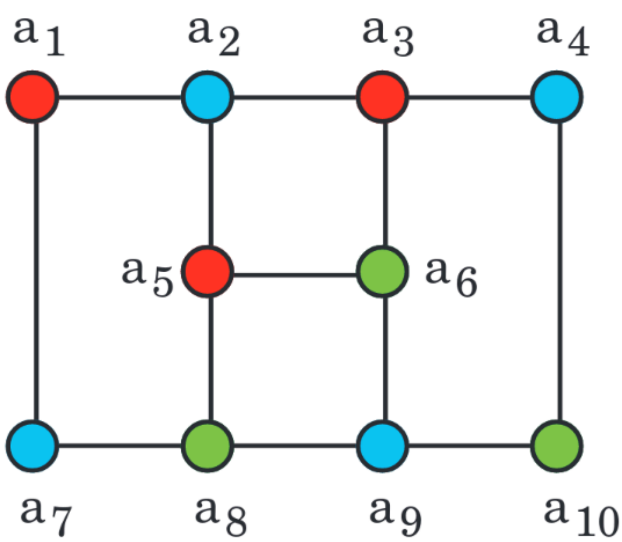
1. ϕ is a distance-1 coloring
2. every **cycle** uses at least 3 colors

Theorem 4.6 Let A be a symmetric matrix with **nonzero diagonal elements**, G be its adjacency matrix. A mapping ϕ is an acyclic coloring of the adjacency graph iff it induces a substitutable partition of the columns of A.

Name acyclic coloring comes from the fact that the subgraph induced by any pair of colors is a forest.

Illustrative example



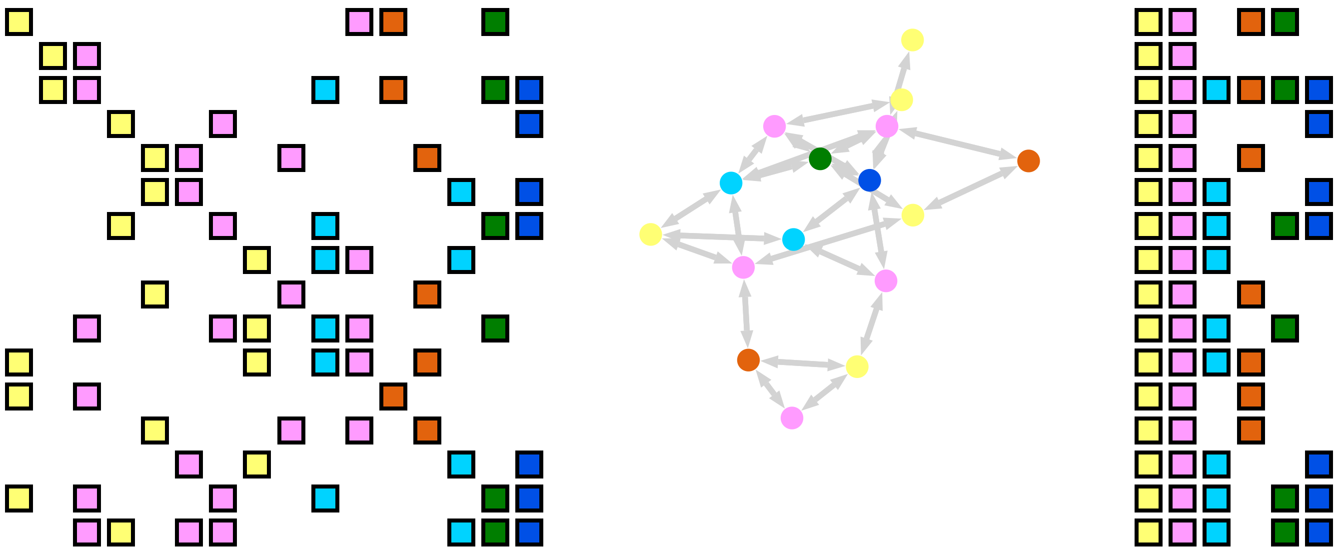


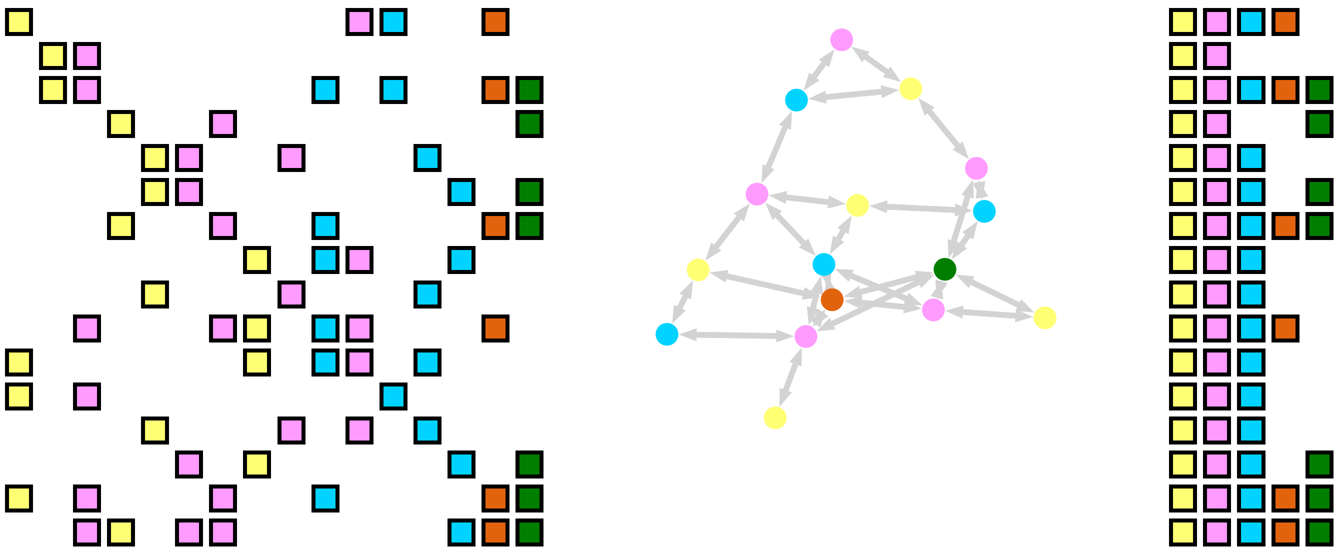
Red-Blue subgraph

$$\begin{pmatrix} a_{21} & a_{23} & a_{25} \\ & a_{43} & \\ & \mathbf{a_{63}} & \mathbf{a_{65}} \\ a_{71} & & \\ & & \mathbf{a_{85}} \end{pmatrix} ; \begin{pmatrix} a_{12} & & a_{17} & \\ a_{32} & a_{34} & & \\ a_{52} & & & \\ & & & \mathbf{a_{69}} \\ & & \mathbf{a_{87}} & \mathbf{a_{89}} \\ & \mathbf{a_{10,4}} & & \mathbf{a_{10,9}} \end{pmatrix}$$

Note: rows with diagonal entries have only one nonzero entries hence have been omitted in the image above.

Larger example : star vs acyclic coloring





n = 16

Comparison of chromatic numbers

Theorem 7.1 For every graph G,

$$\xi_1(G) \leq \xi_{\text{acyclic}}(G) \leq \xi_{\text{star}}(G) \leq \xi_2(G) = \xi_1(G^2)$$

scale = pad = border =

Utils

```
using PlutoUI, PlutoUI.ExperimentalLayout, HypertextLiteral, PlutoTeachingTools
```

```
using SparseArrays, Images, SparseMatrixColorings
```

```
using Graphs, GraphPlot, LinearAlgebra, StableRNGs
```

viz (generic function with 1 method)

```
viz(args...; kws...) = three_columns(colored_plots(args...; kws...)...)
```

colored_plots (generic function with 1 method)

```
function colored_plots(A; decompression, structure, partition = :column, kws...)
    problem = ColoringProblem(; structure, partition)
    algo = GreedyColoringAlgorithm(; decompression)
    result = coloring(A, problem, algo)
    background_color = RGBA(0, 0, 0, 0)
    border_color = RGB(0, 0, 0)
    colorscheme = distinguishable_colors(
        ncolors(result),
        [convert(RGB, background_color), convert(RGB, border_color)];
        dropseed=true,
    )
    A_img, B_img = SparseMatrixColorings.show_colors(result; colorscheme, scale,
    pad, border, background_color, border_color)
    if structure == :symmetric
        S = A
    else
        if partition == :column
            S = A' * A
        else
            S = A * A'
        end
    end
    adj = SparseMatrixColorings.AdjacencyGraph(A)
    gp = gplot(Graphs.SimpleDiGraph(S - Diagonal(diag(S))); nodefillc =
    colorscheme[result.color], kws...)
    A_img, gp, B_img
end
```

img (generic function with 3 methods)

two_columns (generic function with 1 method)

three_columns (generic function with 1 method)