

coloring

November 7, 2023

This file is part of CasADi.

CasADi -- A symbolic framework for dynamic optimization.
Copyright (C) 2010–2023 Joel Andersson, Joris Gillis, Moritz Diehl,
KU Leuven. All rights reserved.
Copyright (C) 2011–2014 Greg Horn

CasADi is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 3 of the License, or (at your option) any later version.

CasADi is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with CasADi; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

```
[1]: from casadi import *
```

Read all about coloring in the seminal paper “What color is your Jacobian?”
<http://www.cs.odu.edu/~pothen/Papers/sirev2005.pdf>

```
[2]: def color(A):  
    print("="*80)  
    print("Original:")  
    print(repr(DM(A,1)))  
    print("Colored: ")  
    print(repr(DM(A.uni_coloring(),1)))
```

```
[3]: A = Sparsity.diag(5)  
    color(A)
```

```
=====
```

Original:
DM(

```
[[1, 00, 00, 00, 00],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [00, 00, 00, 00, 1]])
```

Colored:

```
DM([1, 1, 1, 1, 1])
```

One direction needed to capture all

```
[4]: color(Sparsity.dense(5,10))
```

```
=====
```

Original:

DM(

```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
```

Colored:

DM(

```
[[1, 00, 00, 00, 00, 00, 00, 00, 00, 00],
 [00, 1, 00, 00, 00, 00, 00, 00, 00, 00],
 [00, 00, 1, 00, 00, 00, 00, 00, 00, 00],
 [00, 00, 00, 1, 00, 00, 00, 00, 00, 00],
 [00, 00, 00, 00, 1, 00, 00, 00, 00, 00],
 [00, 00, 00, 00, 00, 1, 00, 00, 00, 00],
 [00, 00, 00, 00, 00, 00, 1, 00, 00, 00],
 [00, 00, 00, 00, 00, 00, 00, 1, 00, 00],
 [00, 00, 00, 00, 00, 00, 00, 00, 1, 00],
 [00, 00, 00, 00, 00, 00, 00, 00, 00, 1]])
```

We need 5 directions. The colored response reads: each row corresponds to a direction; each column correspond to a row of the original matrix.

```
[5]: color(A+Sparsity.triplet(5,5,[0],[4]))
```

```
=====
```

Original:

DM(

```
[[1, 00, 00, 00, 1],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [00, 00, 00, 00, 1]])
```

Colored:

DM(

```
[[1, 00],
```

```
[1, 00],
[1, 00],
[1, 00],
[00, 1]])
```

First 4 rows can be taken together, the fifth row is taken separately

```
[6]: color(A+Sparsity.triplet(5,5,[4],[0]))
```

```
=====
Original:
DM(
[[1, 00, 00, 00, 00],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [1, 00, 00, 00, 1]])
Colored:
DM(
[[1, 00],
 [1, 00],
 [1, 00],
 [1, 00],
 [00, 1]])
```

First 4 rows can be taken together, the fifth row is taken separately

```
[7]: color(A+Sparsity.triplet(5,5,[0]*5,list(range(5))))
```

```
=====
Original:
DM(
[[1, 1, 1, 1, 1],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [00, 00, 00, 00, 1]])
Colored:
DM(
[[1, 00, 00, 00, 00],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [00, 00, 00, 00, 1]])
```

The first row is taken separately. The remaining rows are lumped together in one direction.

```
[8]: color(A+Sparsity.triplet(5,5,list(range(5)),[0]*5))
```

```
=====
```

```
Original:
DM(
[[1, 00, 00, 00, 00],
 [1, 1, 00, 00, 00],
 [1, 00, 1, 00, 00],
 [1, 00, 00, 1, 00],
 [1, 00, 00, 00, 1]])
```

```
Colored:
DM(
[[1, 00],
 [00, 1],
 [00, 1],
 [00, 1],
 [00, 1]])
```

We need 5 directions.

Next, we look at `star_coloring`

```
[9]: def color(A):
      print("="*80)
      print("Original:")
      print(repr(DM(A,1)))
      print("Star colored: ")
      print(repr(DM(A.star_coloring(1),1)))
```

```
[10]: color(A)
```

```
=====
Original:
DM(
[[1, 00, 00, 00, 00],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [00, 00, 00, 00, 1]])
Star colored:
DM([1, 1, 1, 1, 1])
```

One direction needed to capture all

```
[11]: color(Sparsity.dense(5,5))
```

```
=====
Original:
DM(
[[1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
```

```

    [1, 1, 1, 1, 1]])
Star colored:
DM(
[[1, 00, 00, 00, 00],
 [00, 1, 00, 00, 00],
 [00, 00, 1, 00, 00],
 [00, 00, 00, 1, 00],
 [00, 00, 00, 00, 1]])

```

We need 5 directions.

```

[12]: color(A+Sparsity.triplet(5,5,[0]*5,list(range(5)))+Sparsity.
      ↪triplet(5,5,list(range(5)),[0]*5))

```

```


=====
Original:
DM(
[[1, 1, 1, 1, 1],
 [1, 1, 00, 00, 00],
 [1, 00, 1, 00, 00],
 [1, 00, 00, 1, 00],
 [1, 00, 00, 00, 1]])
Star colored:
DM(
[[1, 00],
 [00, 1],
 [00, 1],
 [00, 1],
 [00, 1]])

```

The first row/col is taken separately. The remaining rows/cols are lumped together in one direction.

Let's take an example from the paper

```

[13]: A = 
      ↪DM([[1,1,0,0,0,0],[1,1,1,0,1,1],[0,1,1,1,0,0],[0,0,1,1,0,1],[0,1,0,0,1,0],[0,1,0,1,0,1]])
      A = sparsify(A)
      color(A.sparsity())

```

```

=====
Original:
DM(
[[1, 1, 00, 00, 00, 00],
 [1, 1, 1, 00, 1, 1],
 [00, 1, 1, 1, 00, 00],
 [00, 00, 1, 1, 00, 1],
 [00, 1, 00, 00, 1, 00],
 [00, 1, 00, 1, 00, 1]])
Star colored:

```

```
DM(  
  [[00, 1, 00],  
   [1, 00, 00],  
   [00, 1, 00],  
   [00, 00, 1],  
   [00, 1, 00],  
   [00, 1, 00]])
```