Minimizing Operations in Matrix/Vector Multiplication

Michael Wolf

CS 591MH

10/17/2006

Matrix/Vector Multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} r_1^T \\ \hline r_2^T \\ \hline \vdots \\ \hline r_m^T \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} r_1^T x \\ r_2^T x \\ \vdots \\ r_m^T x \end{bmatrix}$$

$$y = Ax$$

Optimization Problem

Objective: To generate code to minimize the number of multiply add pairs (MAPs) in matrix/vector multiplication

e.g. $r_2 = \alpha r_1$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} r_1^T \\ \hline \alpha r_1^T \\ \hline \vdots \\ \hline r_m^T \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} r_1^T x \\ \alpha y_1 \\ \vdots \\ r_m^T x \end{bmatrix}$$

Possible Optimizations – 0 Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$y_3 = 0$ 0 MAPs

Possible Optimizations - 1 NZ Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_1 = 2x_1 \quad \boxed{\text{1 MAP}}$$

Possible Optimizations - 2 NZ Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_2 = 2x_1 + 2x_2 \quad \boxed{\text{2 MAPs}}$$

Possible Optimizations – 3 NZ Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 1 & 1 \\ 2 & 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_4 = 2x_1 + 2x_2 + 2x_3$$ 3 MAPs

Possible Optimizations - Same Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_4 = y_2 \quad \boxed{\text{0 MAPs}}$$

Possible Optimizations – Scalar Multiple Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_2 = 4y_1 \quad \boxed{\text{1 MAP}}$$

Possible Optimizations - Partial Scalar Multiple Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_2 = 4y_4 + 4x_4 \quad \boxed{\text{2 MAPs}}$$

Possible Optimizations – Linear Dependency Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$y_2 = 4y_3 + 4y_4 \quad \boxed{\text{2 MAPs}}$$

WHY?

Application

- Work by Kirby, et al., University of Chicago
- Finite element "Compilers"
  - FIAT (FE gen.), FFC (variational forms -> code)
  - FErari (optimizer)
- Construction of FE Matrices extremely expensive for large unstructured problems, especially for high order bases
- Methods for reducing redundant operations in building stiffness matrices
  - Many local stiffness matrices built
  - Generate code to optimize construction of local stiffness matrices

2D Laplace – Local Stiffness Matrix Assembly

- Elemental bilinear form:

$$
\begin{aligned}
(\nabla u, \nabla v)_e &= \det(J)(\nabla u, \nabla v)_{\hat{e}} \\
&= \det(J)\Bigg[\left(\frac{\partial u}{\partial r}\frac{\partial r}{\partial x}, \frac{\partial v}{\partial r}\frac{\partial r}{\partial x}\right)_{\hat{e}} + \left(\frac{\partial u}{\partial r}\frac{\partial r}{\partial x}, \frac{\partial v}{\partial s}\frac{\partial s}{\partial x}\right)_{\hat{e}} + \left(\frac{\partial u}{\partial s}\frac{\partial s}{\partial x}, \frac{\partial v}{\partial r}\frac{\partial r}{\partial x}\right)_{\hat{e}} + \left(\frac{\partial u}{\partial s}\frac{\partial s}{\partial x}, \frac{\partial v}{\partial s}\frac{\partial s}{\partial x}\right)_{\hat{e}} + \\
&\qquad \left(\frac{\partial u}{\partial r}\frac{\partial r}{\partial y}, \frac{\partial v}{\partial r}\frac{\partial r}{\partial y}\right)_{\hat{e}} + \left(\frac{\partial u}{\partial r}\frac{\partial r}{\partial y}, \frac{\partial v}{\partial s}\frac{\partial s}{\partial y}\right)_{\hat{e}} + \left(\frac{\partial u}{\partial s}\frac{\partial s}{\partial y}, \frac{\partial v}{\partial r}\frac{\partial r}{\partial y}\right)_{\hat{e}} + \left(\frac{\partial u}{\partial s}\frac{\partial s}{\partial y}, \frac{\partial v}{\partial s}\frac{\partial s}{\partial y}\right)_{\hat{e}}\Bigg] \\
&= \det(J)\left[\frac{\partial r}{\partial x}\left(\frac{\partial u}{\partial r}, \frac{\partial v}{\partial r}\right)_{\hat{e}}\frac{\partial r}{\partial x} + \frac{\partial r}{\partial x}\left(\frac{\partial u}{\partial r}, \frac{\partial v}{\partial s}\right)_{\hat{e}}\frac{\partial s}{\partial x} + \frac{\partial s}{\partial x}\left(\frac{\partial u}{\partial s}, \frac{\partial v}{\partial r}\right)_{\hat{e}}\frac{\partial r}{\partial x} + \frac{\partial s}{\partial x}\left(\frac{\partial u}{\partial s}, \frac{\partial v}{\partial s}\right)_{\hat{e}}\frac{\partial s}{\partial x} + \right. \\
&\qquad \left.\frac{\partial r}{\partial y}\left(\frac{\partial u}{\partial r}, \frac{\partial v}{\partial r}\right)_{\hat{e}}\frac{\partial r}{\partial y} + \frac{\partial r}{\partial y}\left(\frac{\partial u}{\partial r}, \frac{\partial v}{\partial s}\right)_{\hat{e}}\frac{\partial s}{\partial y} + \frac{\partial s}{\partial y}\left(\frac{\partial u}{\partial s}, \frac{\partial v}{\partial r}\right)_{\hat{e}}\frac{\partial r}{\partial y} + \frac{\partial s}{\partial y}\left(\frac{\partial u}{\partial s}, \frac{\partial v}{\partial s}\right)_{\hat{e}}\frac{\partial s}{\partial y}\right] \\
&\;\vdots \\
&= \det(J)\left[\frac{\partial r}{\partial x}\left(\mathbf{u}^{\mathbf{T}} D_{rr}\mathbf{v}\right)\frac{\partial r}{\partial x} + \frac{\partial r}{\partial x}\left(\mathbf{u}^{\mathbf{T}} D_{rs}\mathbf{v}\right)\frac{\partial s}{\partial x} + \frac{\partial s}{\partial x}\left(\mathbf{u}^{\mathbf{T}} D_{sr}\mathbf{v}\right)\frac{\partial r}{\partial x} + \frac{\partial s}{\partial x}\left(\mathbf{u}^{\mathbf{T}} D_{ss}\mathbf{v}\right)\frac{\partial s}{\partial x} + \right. \\
&\qquad \left.\frac{\partial r}{\partial y}\left(\mathbf{u}^{\mathbf{T}} D_{rr}\mathbf{v}\right)\frac{\partial r}{\partial y} + \frac{\partial r}{\partial y}\left(\mathbf{u}^{\mathbf{T}} D_{rs}\mathbf{v}\right)\frac{\partial s}{\partial y} + \frac{\partial s}{\partial y}\left(\mathbf{u}^{\mathbf{T}} D_{sr}\mathbf{v}\right)\frac{\partial r}{\partial y} + \frac{\partial s}{\partial y}\left(\mathbf{u}^{\mathbf{T}} D_{ss}\mathbf{v}\right)\frac{\partial s}{\partial y}\right]
\end{aligned}
$$

$$
D_{rr}(i,j) = \left(\frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_j}{\partial r}\right)_{\hat{e}}, \quad D_{rs}(i,j) = \left(\frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_j}{\partial s}\right)_{\hat{e}}, \quad D_{sr}(i,j) = \left(\frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial r}\right)_{\hat{e}}, \quad D_{ss}(i,j) = \left(\frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial s}\right)_{\hat{e}}
$$

2D Laplace – Local Stiffness Matrix Assembly

$$
\begin{aligned}
(\nabla u, \nabla v)_e \quad = \quad & \mathbf{u}^{\mathbf{T}} \left(\det(J) \left[\frac{\partial r}{\partial x} (D_{rr}) \frac{\partial r}{\partial x} + \frac{\partial r}{\partial x} (D_{rs}) \frac{\partial s}{\partial x} + \frac{\partial s}{\partial x} (D_{sr}) \frac{\partial r}{\partial x} + \frac{\partial s}{\partial x} (D_{ss}) \frac{\partial s}{\partial x} + \right.\right. \\
& \left.\left. \frac{\partial r}{\partial y} (D_{rr}) \frac{\partial r}{\partial y} + \frac{\partial r}{\partial y} (D_{rs}) \frac{\partial s}{\partial y} + \frac{\partial s}{\partial y} (D_{sr}) \frac{\partial r}{\partial y} + \frac{\partial s}{\partial y} (D_{ss}) \frac{\partial s}{\partial y} \right] \right) \mathbf{v}
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
S^e \quad = \quad & \det(J) \left[\frac{\partial r}{\partial x} D_{rr} \frac{\partial r}{\partial x} + \frac{\partial r}{\partial x} D_{rs} \frac{\partial s}{\partial x} + \frac{\partial s}{\partial x} D_{sr} \frac{\partial r}{\partial x} + \frac{\partial s}{\partial x} D_{ss} \frac{\partial s}{\partial x} + \right. \\
& \left. \frac{\partial r}{\partial y} D_{rr} \frac{\partial r}{\partial y} + \frac{\partial r}{\partial y} D_{rs} \frac{\partial s}{\partial y} + \frac{\partial s}{\partial y} D_{sr} \frac{\partial r}{\partial y} + \frac{\partial s}{\partial y} D_{ss} \frac{\partial s}{\partial y} \right]
\end{aligned}
}
$$

- Terms not grouped by element dependency
- 8*(# bases)^2 MAPs

$$
\begin{aligned}
S^e \quad = \quad & D_{rr} \left[\det(J) \left(\frac{\partial r}{\partial x} \frac{\partial r}{\partial x} + \frac{\partial r}{\partial y} \frac{\partial r}{\partial y} \right) \right]_e + D_{rs} \left[\det(J) \left(\frac{\partial r}{\partial x} \frac{\partial s}{\partial x} + \frac{\partial r}{\partial y} \frac{\partial s}{\partial y} \right) \right]_e + \\
& D_{sr} \left[\det(J) \left(\frac{\partial s}{\partial x} \frac{\partial r}{\partial x} + \frac{\partial s}{\partial y} \frac{\partial r}{\partial y} \right) \right]_e + D_{ss} \left[\det(J) \left(\frac{\partial s}{\partial x} \frac{\partial s}{\partial x} + \frac{\partial s}{\partial y} \frac{\partial s}{\partial y} \right) \right]_e
\end{aligned}
$$

2D Laplace - Tensor K

$$\begin{aligned} S^e \;=\;\; & D_{rr}\left[\det(J)\left(\frac{\partial r}{\partial x}\frac{\partial r}{\partial x}+\frac{\partial r}{\partial y}\frac{\partial r}{\partial y}\right)\right]_e + D_{rs}\left[\det(J)\left(\frac{\partial r}{\partial x}\frac{\partial s}{\partial x}+\frac{\partial r}{\partial y}\frac{\partial s}{\partial y}\right)\right]_e + \\ & D_{sr}\left[\det(J)\left(\frac{\partial s}{\partial x}\frac{\partial r}{\partial x}+\frac{\partial s}{\partial y}\frac{\partial r}{\partial y}\right)\right]_e + D_{ss}\left[\det(J)\left(\frac{\partial s}{\partial x}\frac{\partial s}{\partial x}+\frac{\partial s}{\partial y}\frac{\partial s}{\partial y}\right)\right]_e \end{aligned}$$

$$\boxed{S^e_{i,j} = \sum_m^2 \sum_n^2 G^e_{m,n} K_{i,j,m,n} = K_{i,j} : G^e}$$

$$G^e = \begin{bmatrix} \left[\det(J)\left(\frac{\partial r}{\partial x}\frac{\partial r}{\partial x}+\frac{\partial r}{\partial y}\frac{\partial r}{\partial y}\right)\right]_e & \left[\det(J)\left(\frac{\partial r}{\partial x}\frac{\partial s}{\partial x}+\frac{\partial r}{\partial y}\frac{\partial s}{\partial y}\right)\right]_e \\ \left[\det(J)\left(\frac{\partial s}{\partial x}\frac{\partial r}{\partial x}+\frac{\partial s}{\partial y}\frac{\partial r}{\partial y}\right)\right]_e & \left[\det(J)\left(\frac{\partial s}{\partial x}\frac{\partial s}{\partial x}+\frac{\partial s}{\partial y}\frac{\partial s}{\partial y}\right)\right]_e \end{bmatrix}$$

$$K_{i,j} = \begin{bmatrix} D_{rr}(i,j) & D_{rs}(i,j) \\ D_{sr}(i,j) & D_{ss}(i,j) \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial \phi_i}{\partial r},\frac{\partial \phi_j}{\partial r}\right)_{\hat{e}} & \left(\frac{\partial \phi_i}{\partial r},\frac{\partial \phi_j}{\partial s}\right)_{\hat{e}} \\ \left(\frac{\partial \phi_i}{\partial s},\frac{\partial \phi_j}{\partial r}\right)_{\hat{e}} & \left(\frac{\partial \phi_i}{\partial s},\frac{\partial \phi_j}{\partial s}\right)_{\hat{e}} \end{bmatrix}$$

2D Laplace - Tensor K

$$K_{i,j} = \begin{bmatrix} \left(\frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_j}{\partial r}\right)_{\hat{e}} & \left(\frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_j}{\partial s}\right)_{\hat{e}} \\ \left(\frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial r}\right)_{\hat{e}} & \left(\frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial s}\right)_{\hat{e}} \end{bmatrix}$$

	$\phi_{0,0}$		$\phi_{1,0}$		$\phi_{0,1}$	
$\phi_{0,0}$	1	1	0	-1	-1	0
	1	1	0	-1	-1	0
$\phi_{1,0}$	0	0	0	0	0	0
	-1	-1	0	1	1	0
$\phi_{0,1}$	-1	-1	0	1	1	0
	0	0	0	0	0	0

K

$$S^e_{i,j} = K_{i,j} : G^e$$

$$S_{1,2} = -G_{1,2} - G_{2,2}$$

p=1

Optimization Problem

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

- Example of tensor K
- 2D Laplace
- Triangles
- p=2
- Each Block a “row”
- Optimize Frobenius product

$$S^e_{i,j} = K_{i,j} : G^e$$

Initial Implementation

- Modified 2D Helmholtz Matlab code from CS598LO
- "Tensor" representation of local stiffness matrix
- Several optimizations (only binary relationships)
  - 0 block
  - Same block
  - 1 NZ, 2NZ, 3NZ
  - Scalar multiple block
  - Partial scalar multiple block
- Graph model for more complex block relationships
- Implementation generates an optimal (minimal MAPs) set of operations to build stiffness matrix
  - Optimal for block relationships used
- Now have C++ matrix/vector multiplication optimization code using FErari tensors

Initial Implementation Example

S =

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Initial Implementation Example – 0 Blocks

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

No change to S

Initial Implementation Example

S =

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Initial Implementation Example – Same Blocks

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

0 MAPs

$S_{i,j} = S_{k,l}$

Initial Implementation Example

S =

0	0	0	0	0	0
0	0	$S(1,2)$	0	0	0
0	$S(2,1)$	0	0	0	0
0	$S(2,4)$	0	$S(2,2)$	0	$S(1,4)$
0	$S(2,5)$	0	$S(4,5)$	$S(2,2)$	$S(3,5)$
0	0	0	$S(4,1)$	$S(5,3)$	0

Initial Implementation Example – Scalar Multiple

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

1 MAP

$$S_{i,j} = \alpha S_{k,l}$$

Initial Implementation Example

$$S = \begin{bmatrix} 0 & 0 & -\frac{1}{4}S(1,2) & 0 & 0 & -\frac{1}{4}S(1,4) \\ 0 & 0 & S(1,2) & 0 & 0 & 0 \\ -\frac{1}{4}S(2,1) & S(2,1) & 0 & 0 & 0 & -\frac{1}{4}S(3,5) \\ 0 & S(2,4) & 0 & S(2,2) & 0 & S(1,4) \\ 0 & S(2,5) & 0 & S(4,5) & S(2,2) & S(3,5) \\ -\frac{1}{4}S(4,1) & 0 & -\frac{1}{4}S(5,3) & S(4,1) & S(5,3) & 0 \end{bmatrix}$$

Initial Implementation Example

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

- Build graph from remaining blocks
- One vertex for each block
- One dot product vertex
- Weighted edges
- Weights – work to determine a block given another block
- Find MST

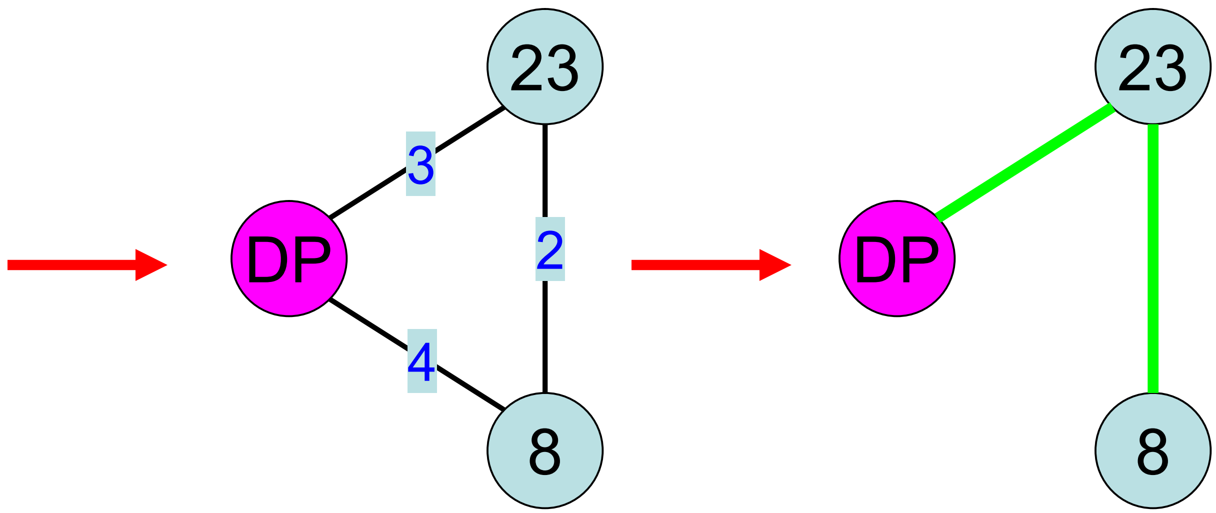
Graph Problem

B8

8	4
4	8

B23

0	-4
-4	-8



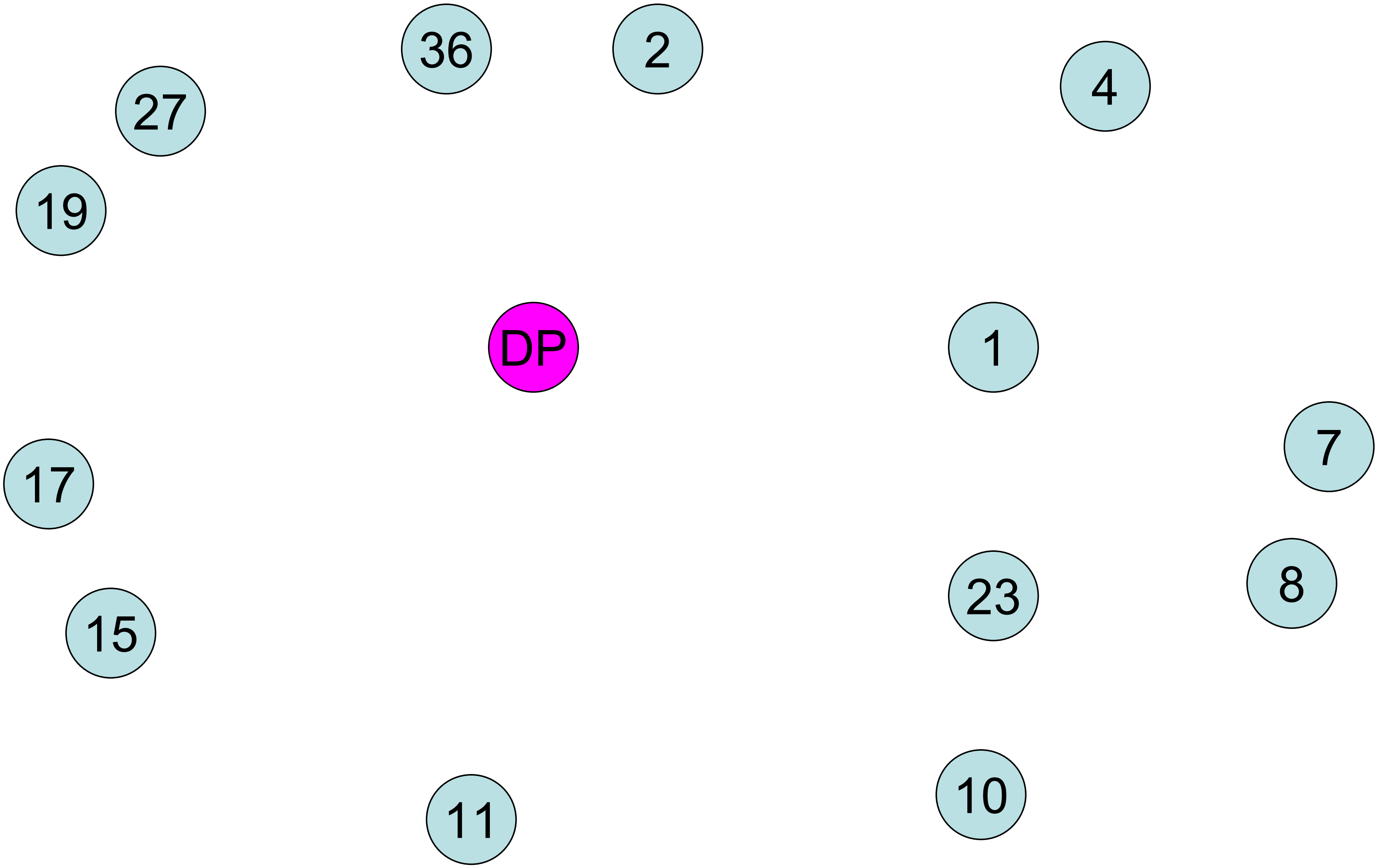
Graph MST(5)

$$\boxed{S_8 = -S_{23} + 8G_{1,1}}$$

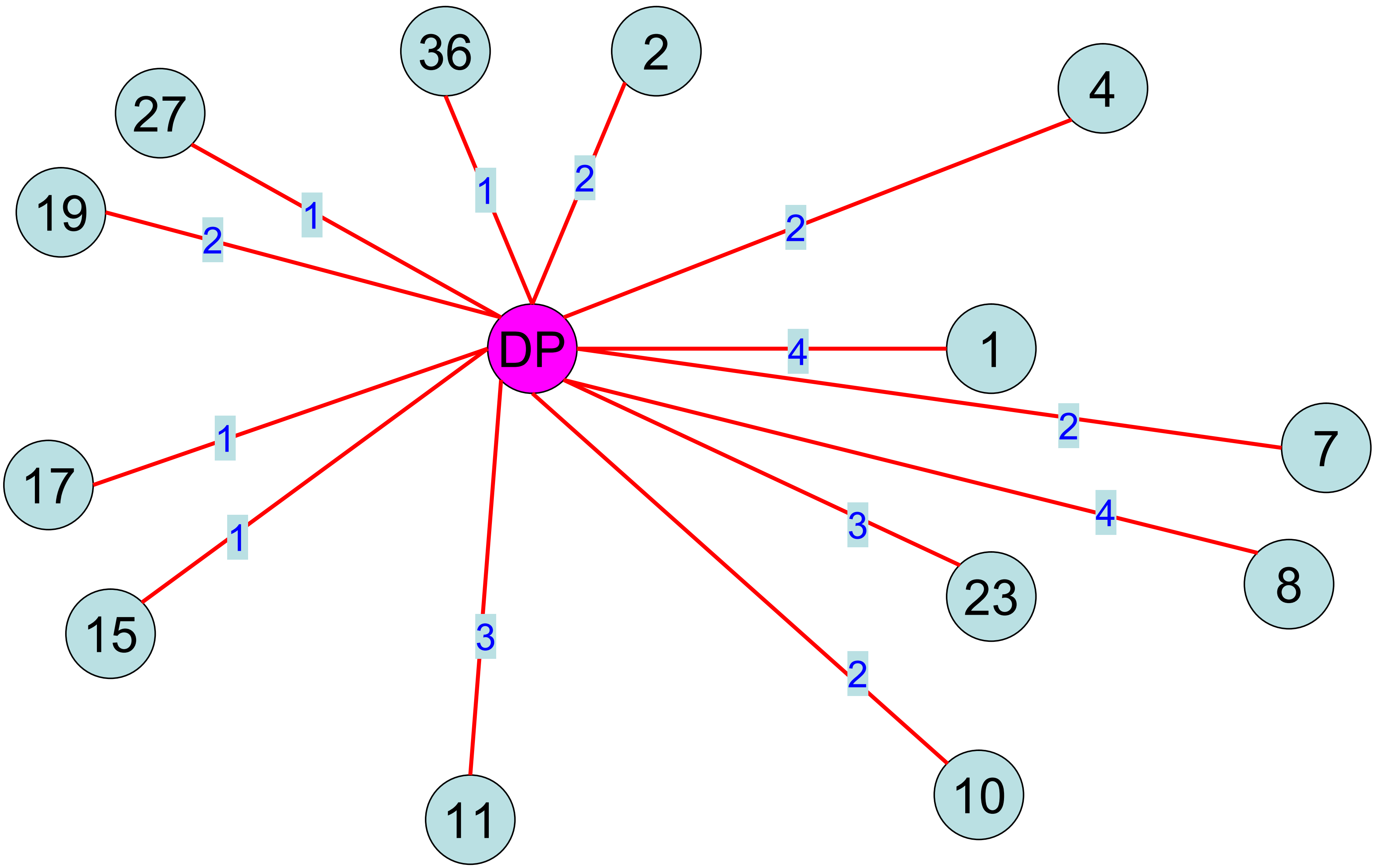
Initial Implementation Example

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

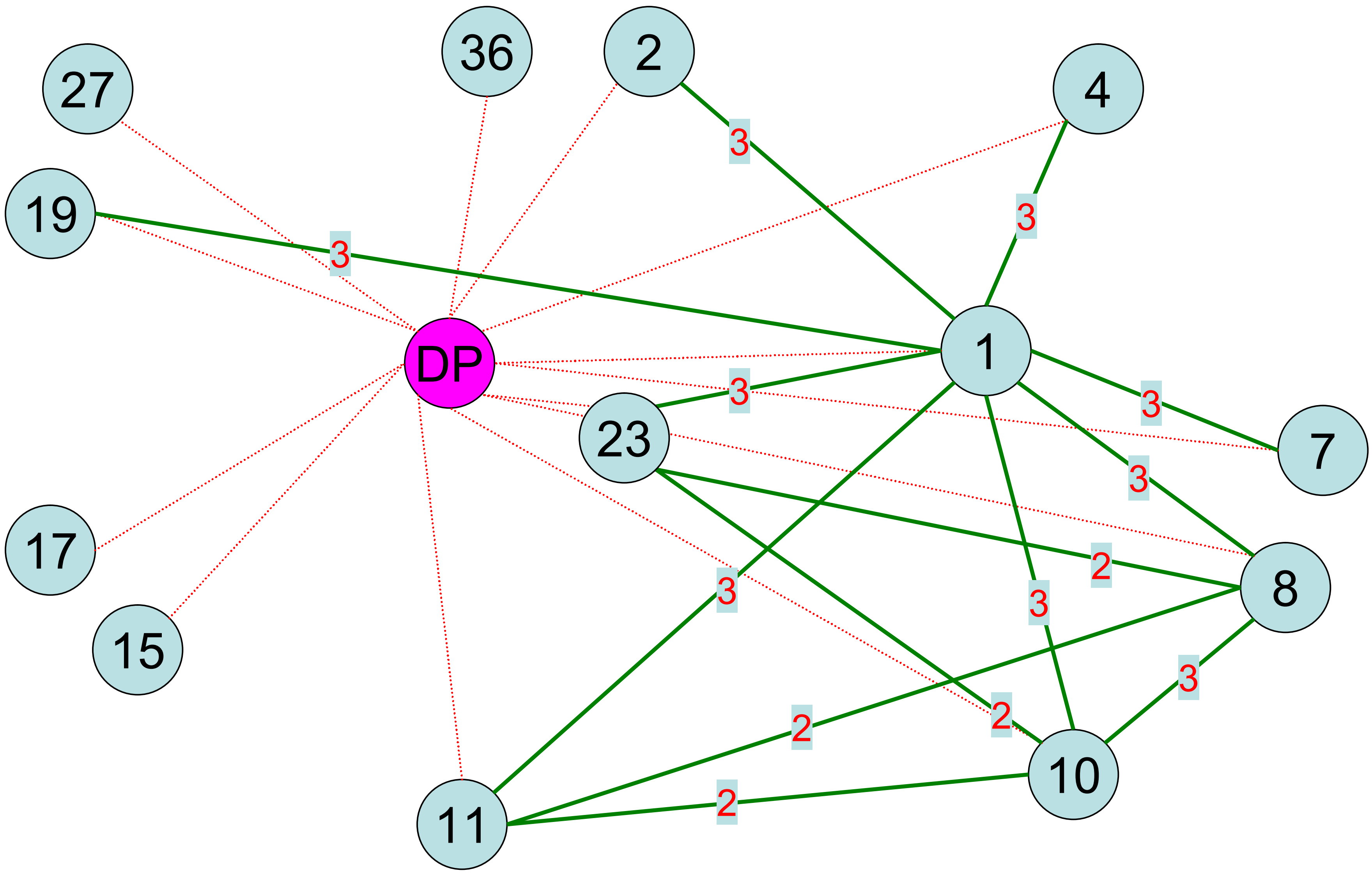
Graph Vertices



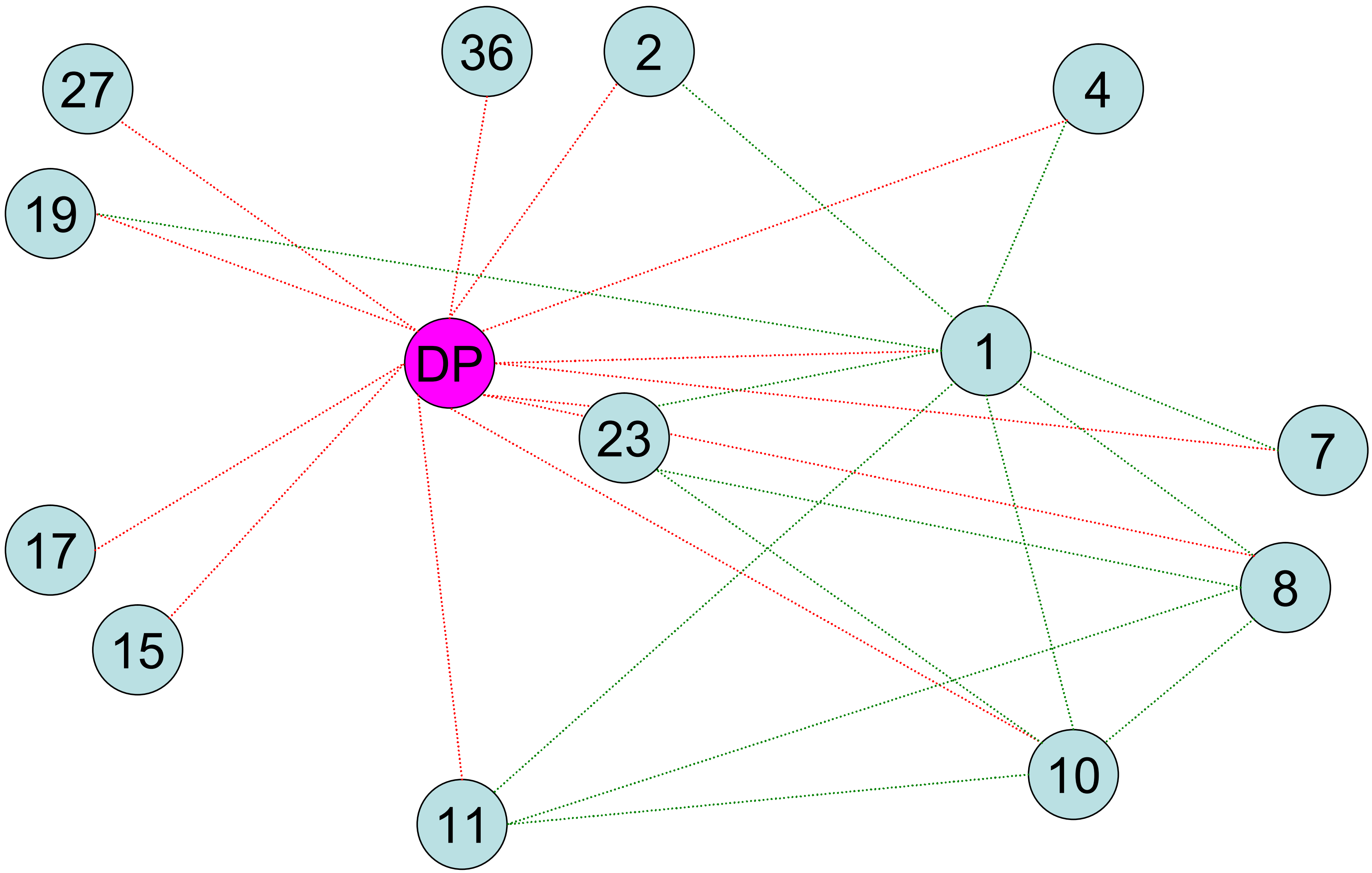
Graph - Dot Product Edges



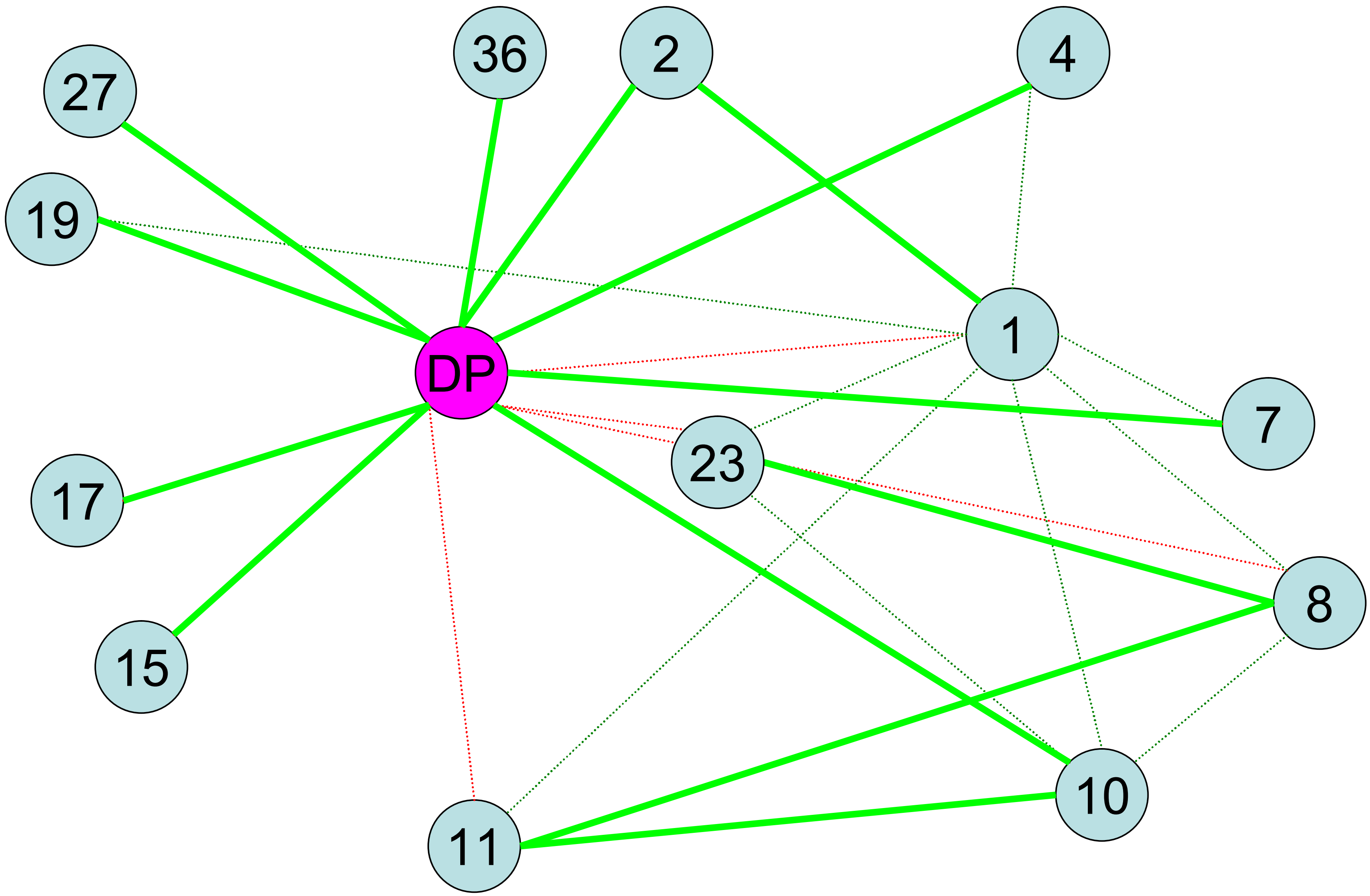
Graph - Partial Scalar Multiple Edges



Graph



Kruskal's Algorithm -> MST



Operations Generated for Each Entry in S

S =

$-\frac{3}{4}S(1,2)$ $+3G(1,1)$ $+3G(2,1)$	$-4G(1,2)$ $-4G(2,2)$	$-\frac{1}{4}S(1,2)$	$-4G(1,1)$ $-4G(2,1)$	0	$-\frac{1}{4}S(1,4)$
$-4G(2,1)$ $-4G(2,2)$	$-S(2,5)$ $+8G(2,2)$	$S(1,2)$	$4G(1,2)+$ $4G(2,1)$	$-S(2,4)$ $-8G(1,1)$	0
$-\frac{1}{4}S(2,1)$	$S(2,1)$	$3G(2,2)$	0	$4G(2,1)$	$-\frac{1}{4}S(3,5)$
$-4G(1,1)$ $-4G(1,2)$	$S(2,4)$	0	$S(2,2)$	$-S(2,2)$ $+8G(1,1)$	$S(1,4)$
0	$S(2,5)$	$4G(1,2)$	$S(4,5)$	$S(2,2)$	$S(3,5)$
$-\frac{1}{4}S(4,1)$	0	$-\frac{1}{4}S(5,3)$	$S(4,1)$	$S(5,3)$	$3G(1,1)$

144 MAPs

$\downarrow$

29 MAPs

Code Generation

$$S(1,2) = -4*G(1,2) - 4*G(2,2);$$
$$S(1,4) = -4*G(1,1) - 4*G(2,1);$$
$$S(2,1) = -4*G(2,1) - 4*G(2,2);$$
$$S(2,4) = 4*G(1,2) + 4*G(2,1);$$
$$S(3,3) = 3*G(2,2);$$
$$S(3,5) = 4*G(2,1);$$
$$S(4,1) = -4*G(1,1) - 4*G(1,2);$$
$$S(5,3) = 4*G(1,2);$$
$$S(6,6) = 3*G(1,1);$$
$$S(1,1) = -0.75*S(1,2) + 3*G(1,1) + 3*G(2,1);$$
$$S(2,5) = -1*S(2,4) - 8*G(1,1);$$
$$S(2,2) = -1*S(2,5) + 8*G(2,2);$$
$$S(4,5) = -1*S(2,2) + 8*G(1,1);$$
$$S(1,3) = -0.25*S(1,2);$$
$$S(1,6) = -0.25*S(1,4);$$

$$S(3,1) = -0.25*S(2,1);$$
$$S(3,6) = -0.25*S(3,5);$$
$$S(6,1) = -0.25*S(4,1);$$
$$S(6,3) = -0.25*S(5,3);$$
$$S(2,3) = S(1,2);$$
$$S(3,2) = S(2,1);$$
$$S(4,2) = S(2,4);$$
$$S(4,4) = S(2,2);$$
$$S(4,6) = S(1,4);$$
$$S(5,2) = S(2,5);$$
$$S(5,4) = S(4,5);$$
$$S(5,5) = S(2,2);$$
$$S(5,6) = S(3,5);$$
$$S(6,4) = S(4,1);$$
$$S(6,5) = S(5,3);$$

Initial Results

Order	Entries	Base MAPs	Opt. MAPs
1	9	36	15(14)
2	36	144	29(28)
3	100	400	155
4	225	900	443
5	441	1764	814
6	784	3136	1387
7	1296	5184	2211

- Greatly reduced MAPs when building stiffness matrix
- Reduction for lower order FE simple
  - Many zeros in K
- Reduction for higher order FE more interesting
  - Fewer zeros in K
  - More complex algorithms necessary

Limitation of Graph Model

3	3	0	-4	0	1	-4	0	0	0	1	0
3	3	0	-4	0	1	-4	0	0	0	1	0
0	0	8	4	0	-4	0	4	-8	-4	0	0
-4	-4	4	8	0	-4	4	0	-4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	-4	-4	0	3	0	0	4	0	-1	0
-4	-4	0	4	0	0	8	4	0	-4	-4	0
0	0	4	0	0	0	4	8	-4	-8	-4	0
0	0	-8	-4	0	4	0	-4	8	4	0	0
0	0	-4	0	0	0	-4	-8	4	8	4	0
1	1	0	0	0	-1	-4	-4	0	4	3	0
0	0	0	0	0	0	0	0	0	0	0	0

$$S_{1,1} = 3S_{1,3} + 3S_{1,6}$$

2 MAPs

- Edges connect 2 vertices
- Can exploit only binary block relationships
- Need hypergraphs with hyperedges

Linear Dependence Hyperedge Representation

$$S_1 = 3S_3 + 3S_6$$

B1

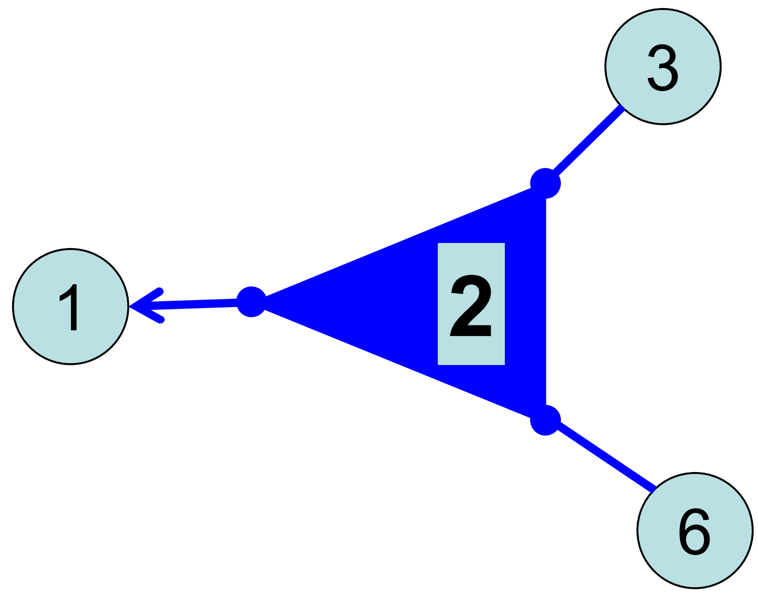
3	3
3	3

B3

0	1
0	1

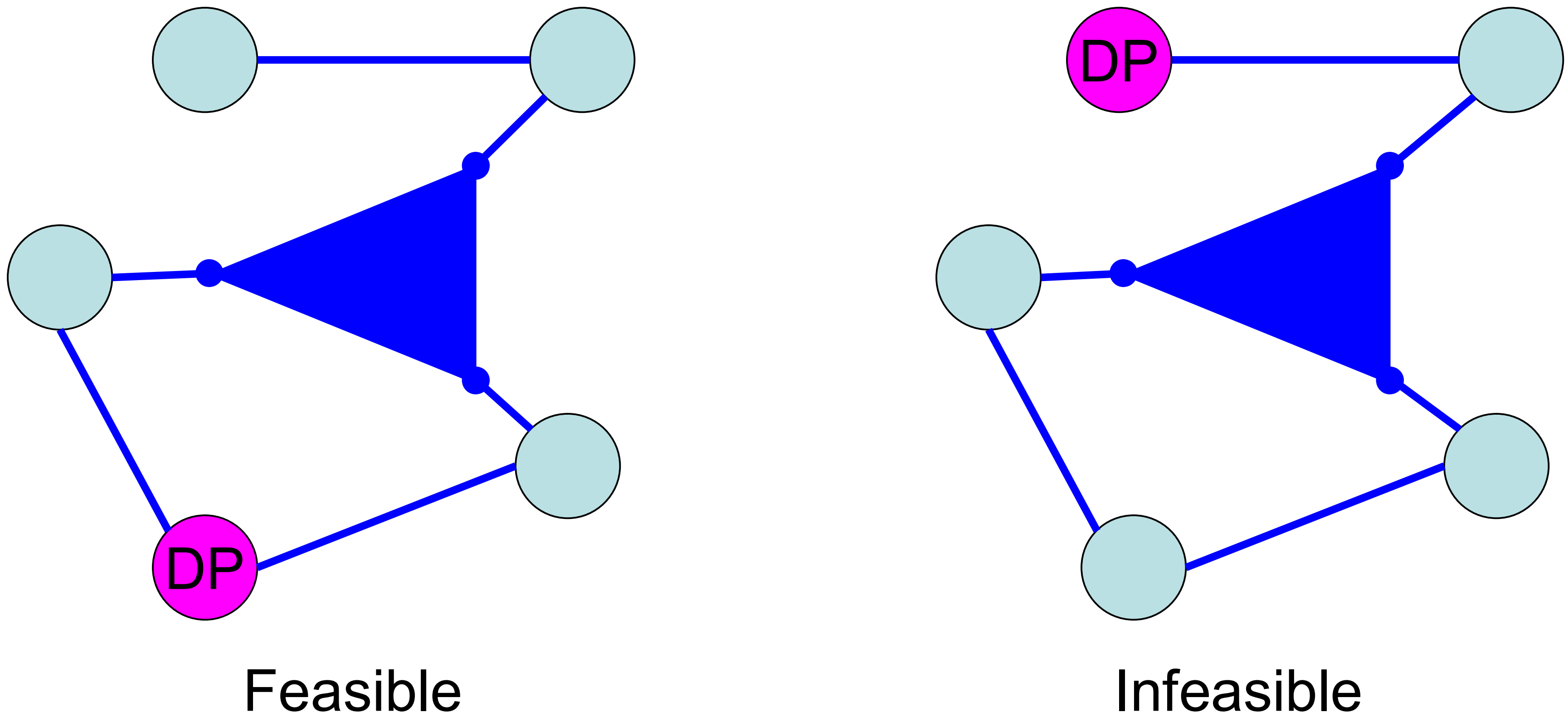
B6

1	0
1	0



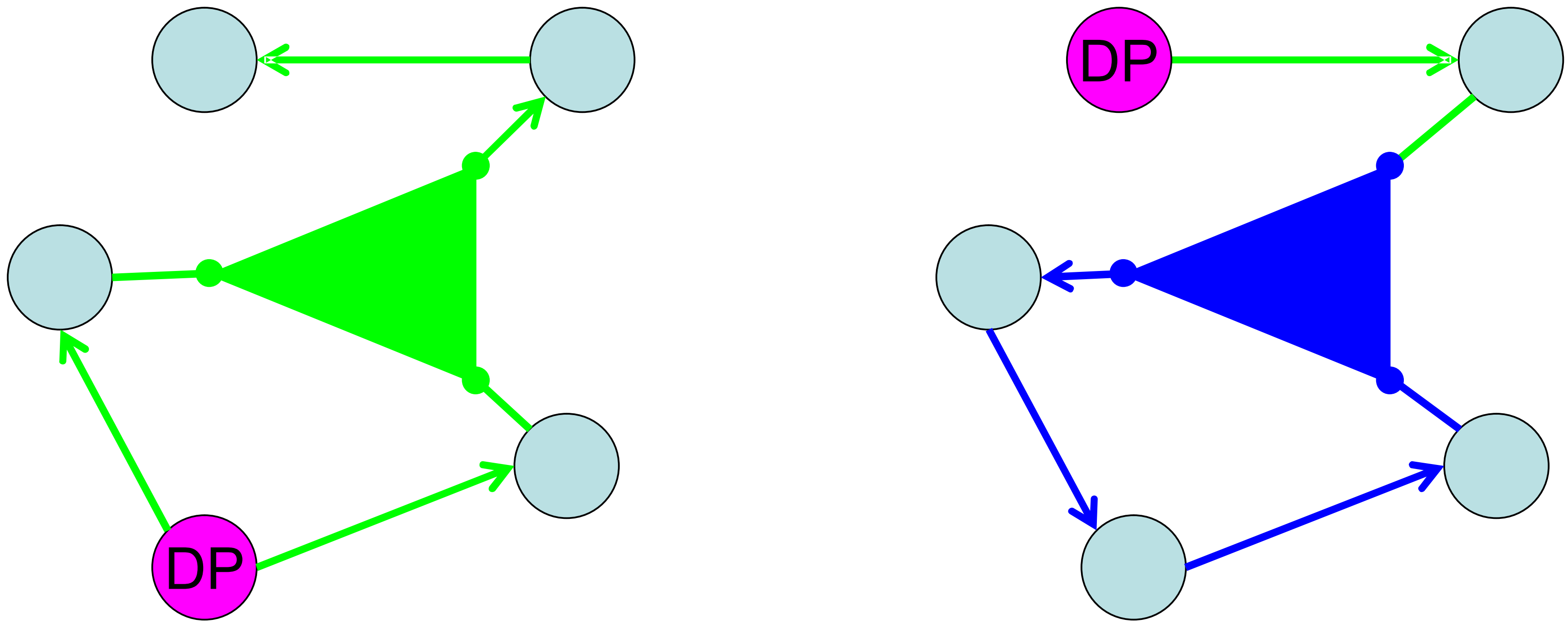
Directed Hyperedge

Argument for Directed Hyperedge Representation



Undirected – almost the same graph

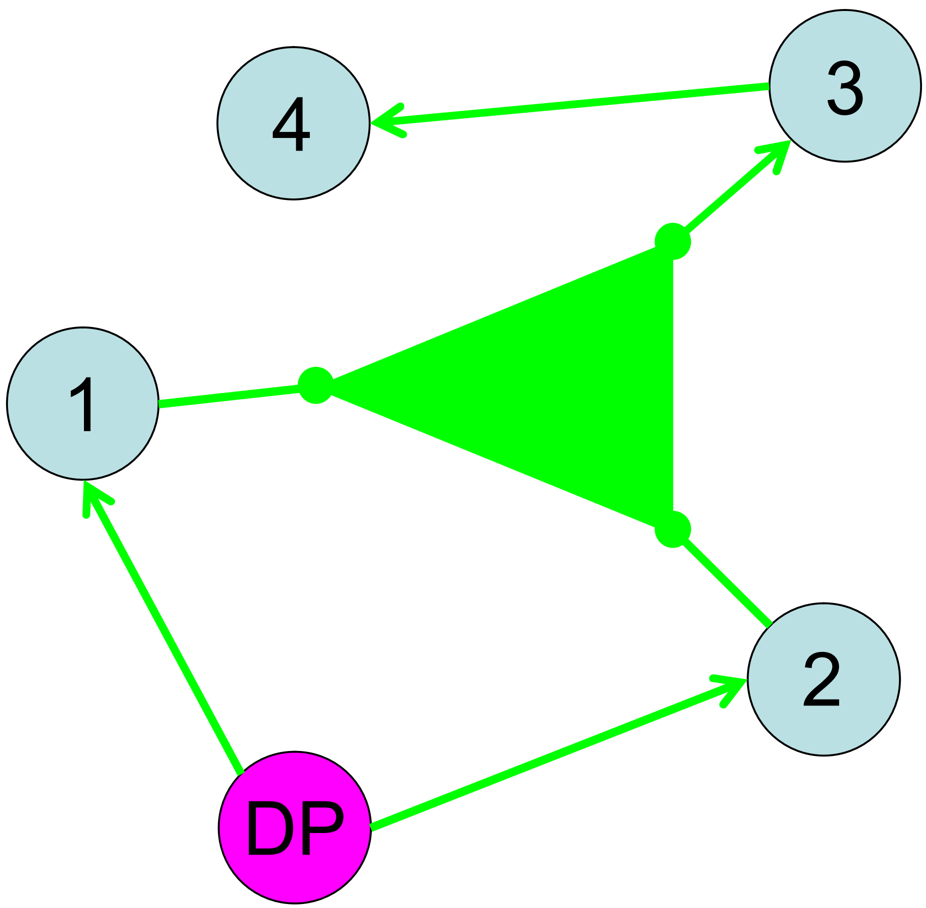
Argument for Directed Hyperedge Representation



Feasible　　　　Infeasible

- Directed – clearly different graphs
- Disadvantage – more edges needed

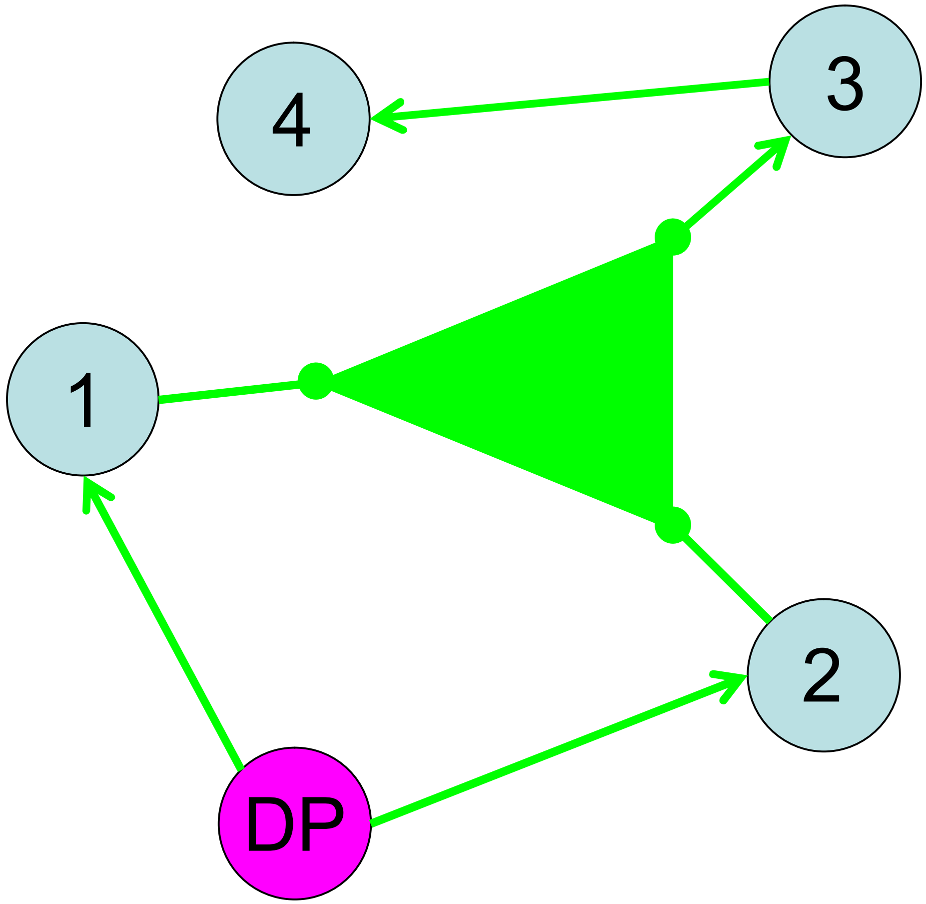
Directed Hypergraph Problem



Feasible

- How do we characterize feasible solutions?
- Solution not necessarily MST

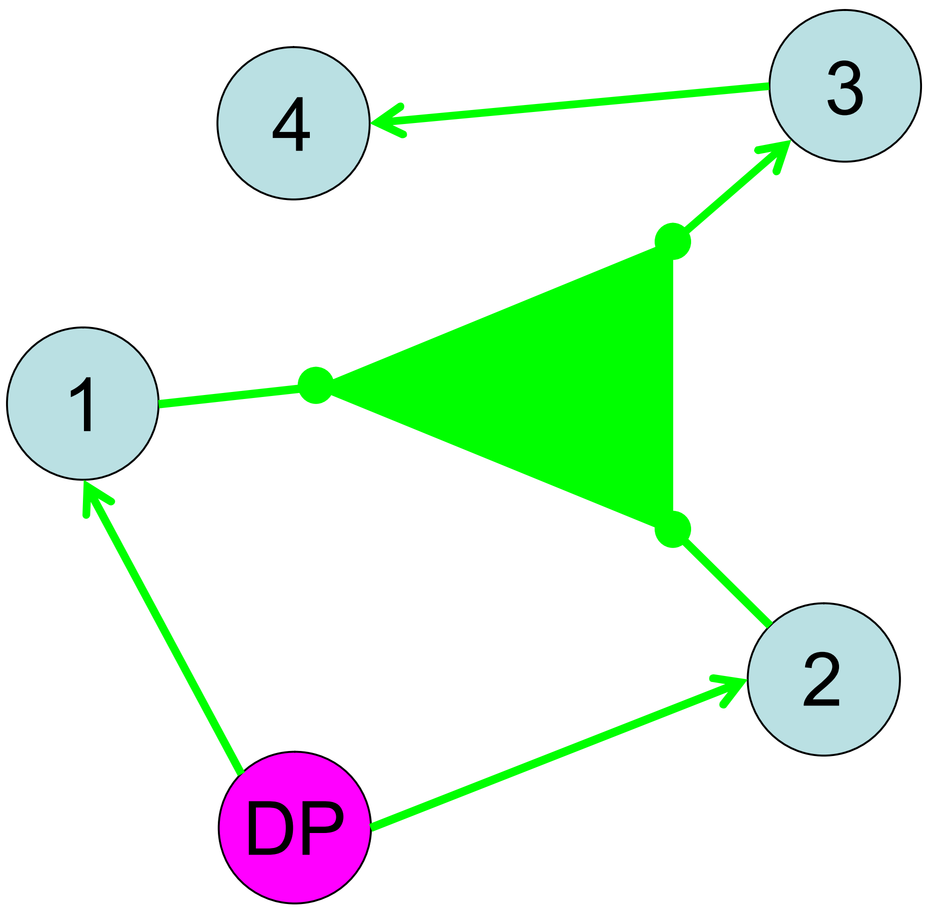
Feasible Directed Hypergraph Solution



Feasible

Every vertex reachable via valid path from DP

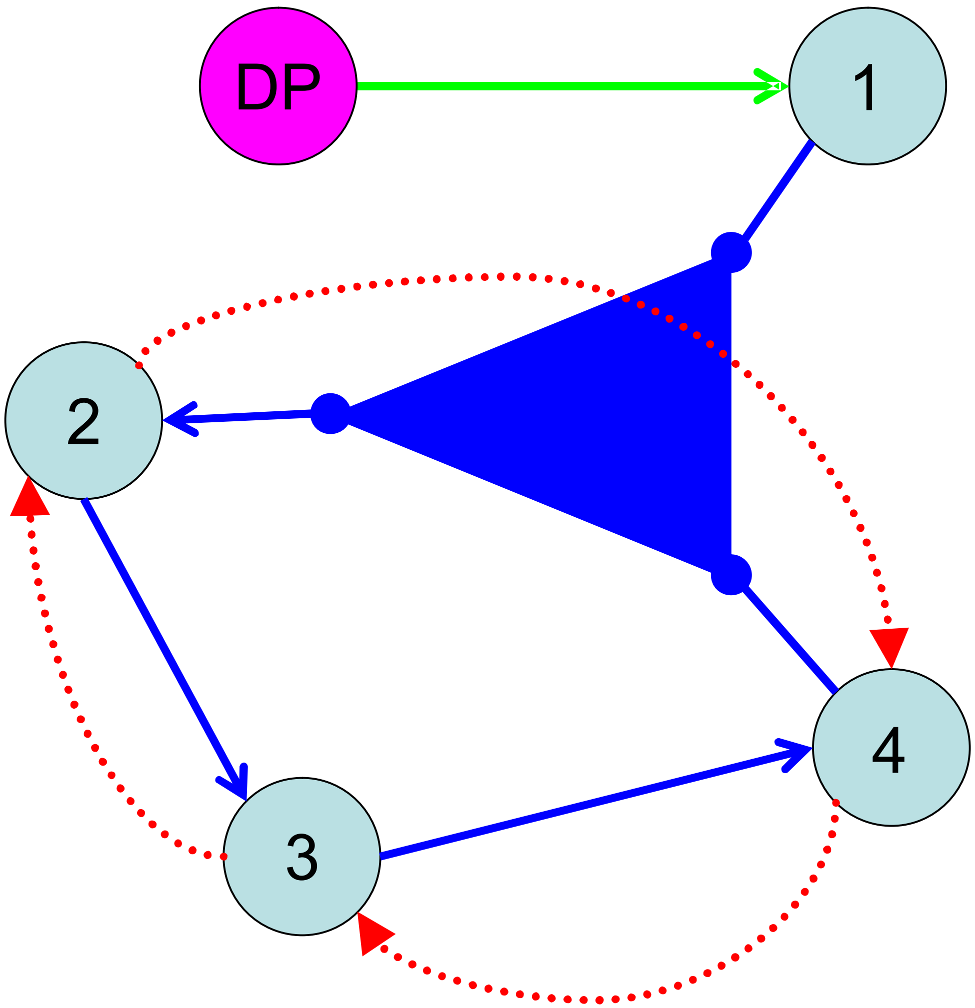
Feasible Directed Hypergraph Solution



Feasible

- Graph Connected
- All vertices are sinks except for DP
- No dependency cycles

Dependency Cycle



Combinatorial Optimization Problem

$$\min c^T x_e \longleftarrow$$ Objective function

$c \longleftarrow$ Vector of edge costs

$x_e \longleftarrow$ Vector of binary variables

- Mathematical formulation into combinatorial optimization problem
- Use IP software to solve this problem
- Optimal solution when IP problem solved exactly
- IP is NP-hard

Notation

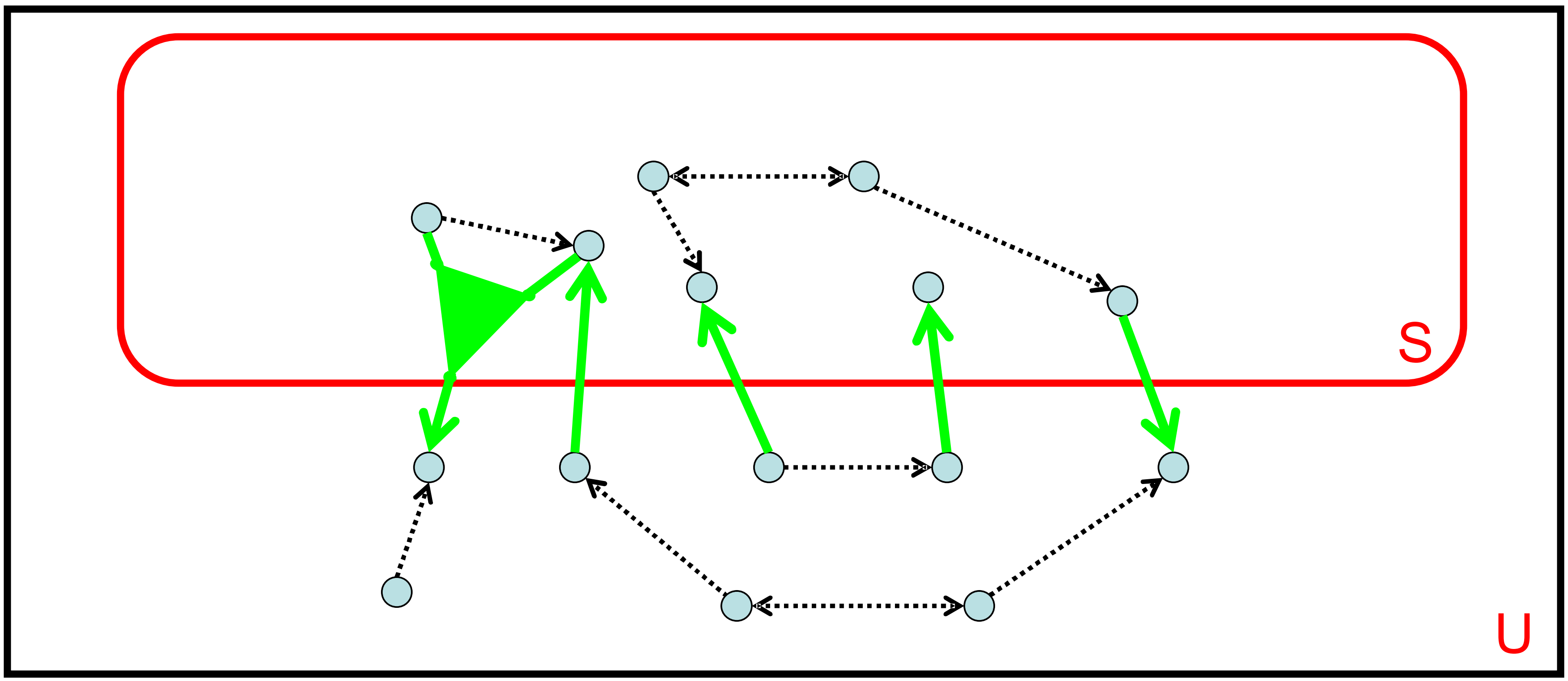
$$X(F) \quad := \quad \sum_{e \in F} x_e$$

Set of edges

Notation

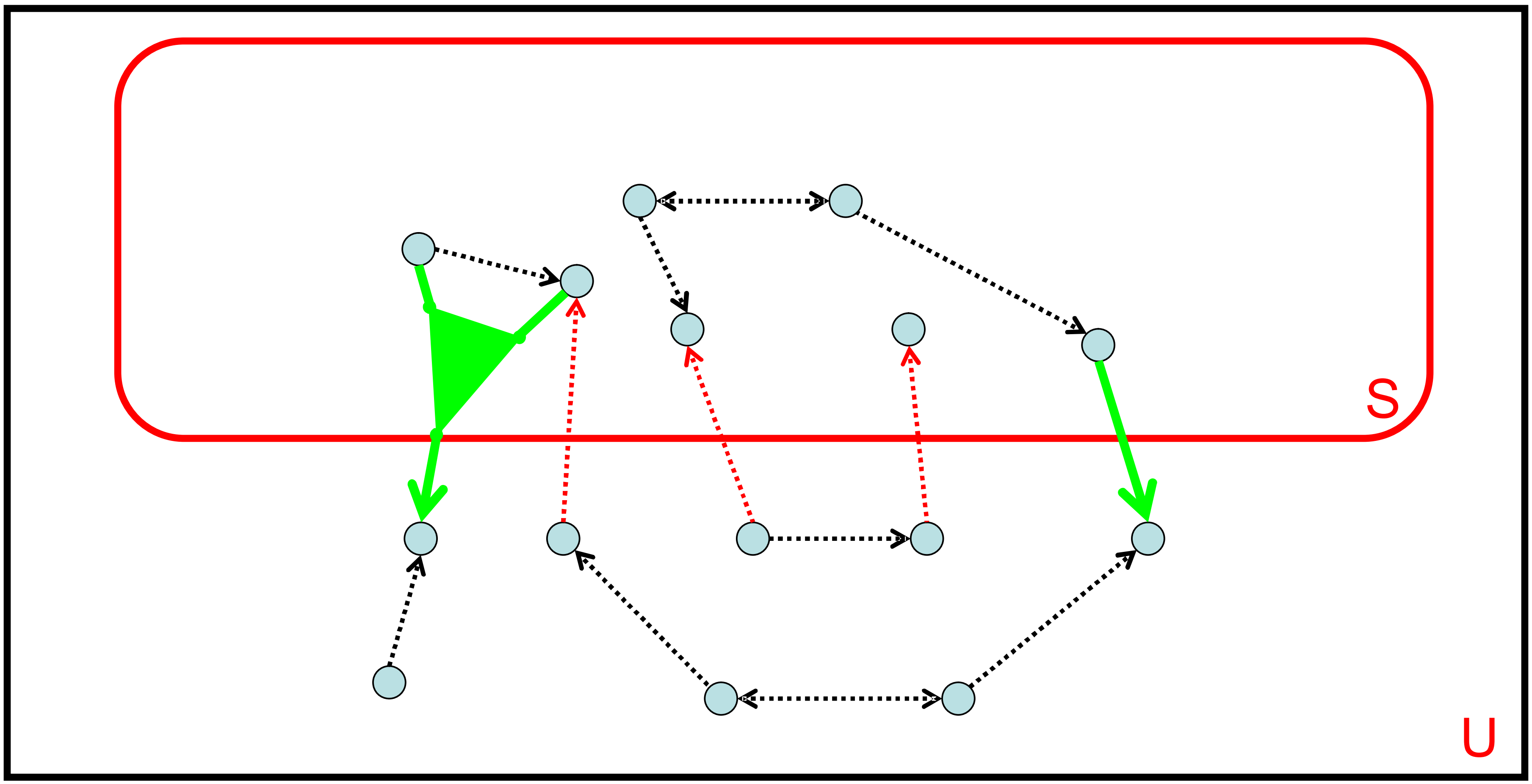
$$\delta(S) \quad := \quad \{e \in E : V_e \cap S \neq \emptyset,\ V_e \cap \bar{\mathrm{S}} \neq \emptyset\}$$

Set of vertices



Notation

$$\delta^-(S) \quad := \quad \{e \in E : V_e \cap S \neq \emptyset, V_e \cap \bar{\mathrm{S}} = sink(e)\}$$



Combinatorial Optimization Problem

$$
\begin{aligned}
&\min c^T x_e \\
&s.t. \\
&\qquad \textstyle\sum_{e:v=sink(e)} x_e = 1,\ \forall v \neq v_{DP} \\
&\qquad X(\delta^{-}(S)) \geq 1, \forall S \subset V : v_{DP} \in S
\end{aligned}
$$

Combinatorial Optimization Problem

$$\min c^T x_e$$

$$s.t.$$

All vertices sinks except for DP

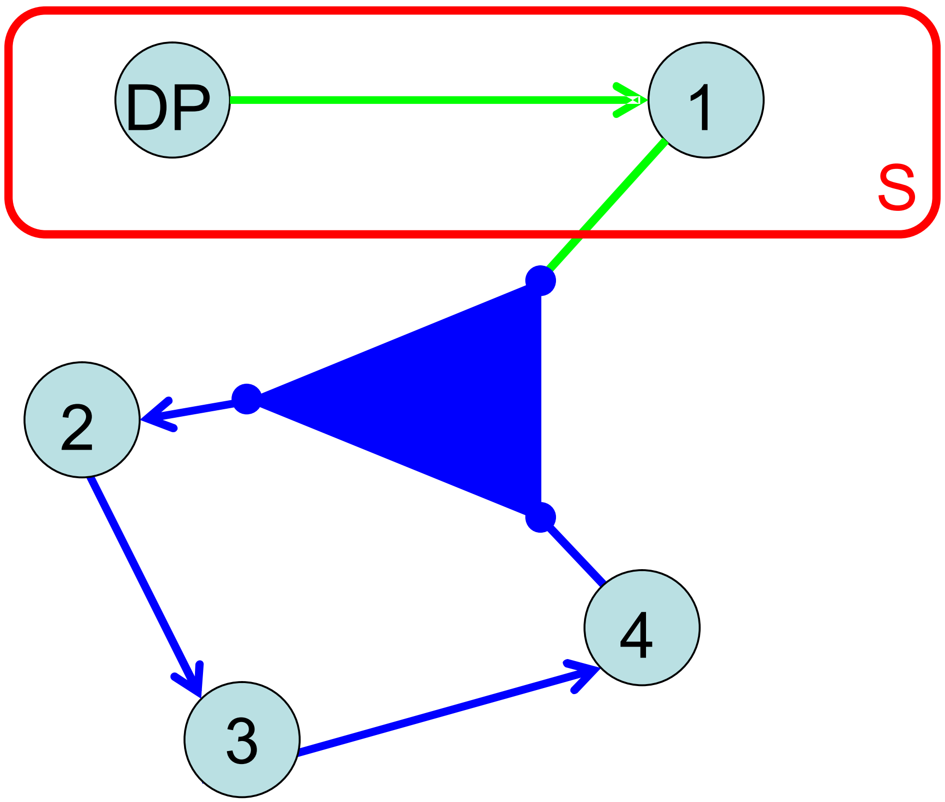
$$\sum_{e:v=sink(e)} x_e = 1, \ \forall v \neq v_{DP}$$

$$X(\delta^-(S)) \geq 1, \forall S \subset V : v_{DP} \in S$$

Connectivity, no dependency cycles

Example 1: Applying 2nd Condition

$$X(\delta^-(S)) \geq 1, \forall S \subset V : v_{DP} \in S$$

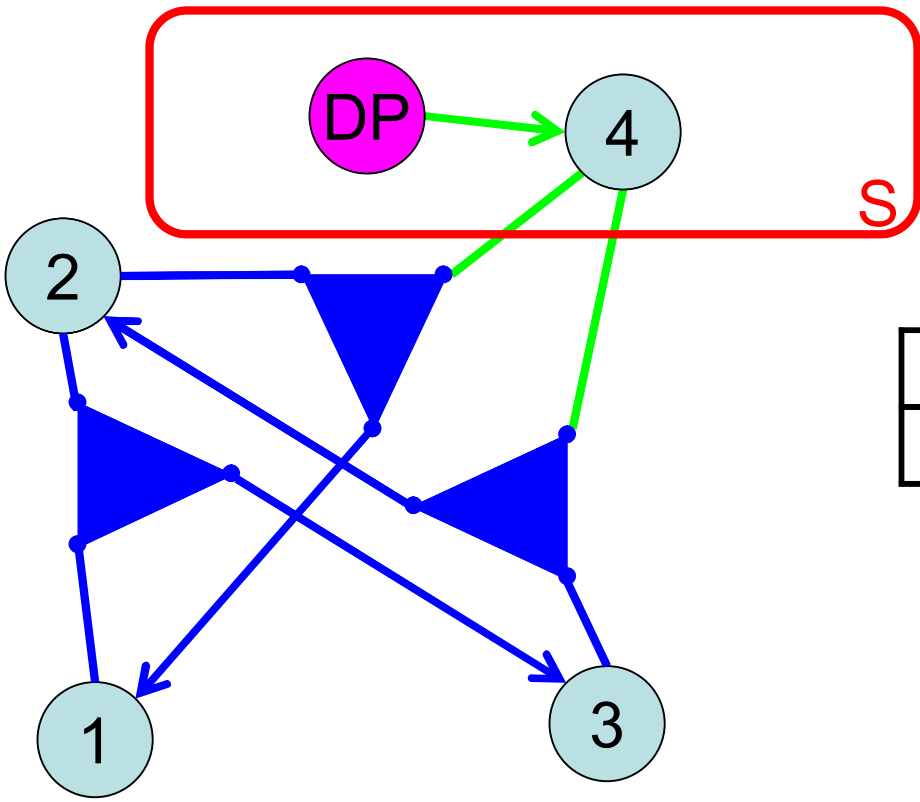


Infeasible

S	$e \in \delta^-(S) : x_e = 1$
$\{DP, 1\}$	✗

Example 2: Applying 2nd Condition

$$X(\delta^-(S)) \geq 1, \forall S \subset V : v_{DP} \in S$$

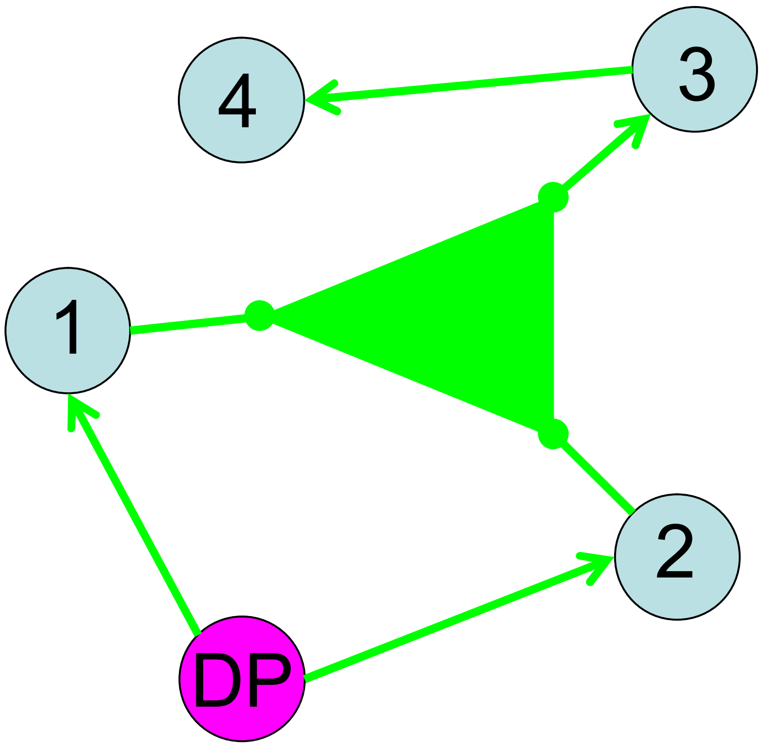


Infeasible

S	$e \in \delta^-(S) : x_e = 1$
$\{DP, 4\}$	X

Example 3: Applying 2nd Condition

$$X(\delta^-(S)) \geq 1, \forall S \subset V : v_{DP} \in S$$



Feasible

S	$e \in \delta^-(S) : x_e = 1$
$\{DP\}$	$\{DP, 1\}$
$\{DP, 1\}$	$\{DP, 2\}$
$\{DP, 2\}$	$\{DP, 1\}$
$\{DP, 3\}$	$\{DP, 1\}$
$\{DP, 4\}$	$\{DP, 1\}$
$\{DP, 1, 2\}$	$\{1, 2, 3\}$
$\{DP, 1, 3\}$	$\{DP, 2\}$
$\{DP, 1, 4\}$	$\{DP, 2\}$
$\{DP, 2, 3\}$	$\{DP, 1\}$
$\{DP, 2, 4\}$	$\{DP, 1\}$
$\{DP, 3, 4\}$	$\{DP, 1\}$
$\{DP, 1, 2, 3\}$	$\{3, 4\}$
$\{DP, 1, 2, 4\}$	$\{1, 2, 3\}$
$\{DP, 1, 3, 4\}$	$\{DP, 2\}$
$\{DP, 2, 3, 4\}$	$\{DP, 1\}$

Acknowledgements

Boman, Erik. Unpublished notes and correspondence. SNL. 2005.

Kirby, R. C. (2006). FErari (version 0.1.0), University of Chicago.

Kirby, R. C., Anders Logg, L. Ridgeway Scott, Andy R. Terrel (2005). "Topological optimization of the evaluation of finite element matrices." University of Chicago Technical Reports: 1-22.

Kirby, R. C., Matthew Knepley, Anders Logg, L. Ridgeway Scott (2005). "Optimizing the Evaluation of Finite Element Matrices." SIAM Journal on Scientific Computing 27(no 3): 741-758.

Professor Olson, et al. 2D Helmholtz code.