Assiomi

Sia $\mathbb{T}^i = \{T_1^i, T_2^i, \dots, T_{N_i}^i\}$ l'insieme delle variabili temporali del $\mathbf{MatrixMult}$ $i-$esimo

Sia $\mathbb{T}_{\text{reg}}^i \subset \mathbb{T}^i$ l'insieme delle variabili temporali storate in registri nel $\mathbf{MM}_i$
e sia $\mathbb{T}_{\text{mem}}^i$ l'insieme delle variabili temporali storate in memoria.
Si noti che $\mathbb{T}_{\text{reg}}^i \cup \mathbb{T}_{\text{mem}}^i = \mathbb{T}^i$, $\mathbb{T}_{\text{reg}}^i \cap \mathbb{T}_{\text{mem}}^i = \varnothing$
Chiamiamo $\{\mathbb{T}_{\text{reg}}^i, \mathbb{T}_{\text{mem}}^i\}$ $\textbf{partizione registri memoria}$ e la denotiamo come $\mathcal{P}_{MR}^i$

Contemporaneamente, sia $\mathbb{T}_{in}^i \subset \mathbb{T}^i$ l'insieme delle variabili temporali che contengono l'input del $\mathbf{MM}_i$
e $\mathbb{T}_{out}^i$ quelle dell'output.
Chiamiamo $\{\mathbb{T}_{\text{in}}^i, \mathbb{T}_{\text{out}}^i\}$ $\textbf{partizione input output}$ e la denotiamo come $\mathcal{P}_{IO}^i$

Denotiamo con $\mathbb{M}$ l'insieme degli indirizzi di memoria

Denotiamo con $\mathbb{D}$ l'insieme dei registri

Partizione in $\mathbb{T}_{mem}^i$

Sia $\mathcal{P}_{\substack{FREE\\CONST}}^i = \{T_{\substack{mem\\constr}}^i, T_{\substack{mem\\free}}^i\}$

Mapping

Definiamo una famiglia di funzioni invertibili $\left\{\phi_i^{i+1} : \mathbb{T}_{out}^i \rightarrow \mathbb{T}_{in}^{i+1}\right\}_i$ che associano ad ogni variabile temporale di output, la variabile temporale di input corrispondente nel matrix mult successivo.

Data ϕ_x^y denotiamo come ϕ_y^x la sua inversa

Sia $m_i : \mathbb{T}_{mem}^i \rightarrow \mathbb{M}$ la funzione che associa ogni variabile temporale ad un indirizzo di memoria

Sia $r_i : \mathbb{T}_{reg}^i \rightarrow \mathbb{D}$ la funzione che associa ogni variabile temporale ad un indirizzo di memoria

Supponiamo m_i, r_i nota a priori. (ottimizzazione libera del primo matrixmult)

Vogliamo definire m_{i+1}, r_{i+1} tale che

1. $m_i|_{\substack{T \in \mathbb{T}_{mem}^i \,:\, T \in \mathbb{T}_{out}^i \cap \mathbb{T}_{mem}^i \wedge \\ \phi_i^{i+1} T \in \mathbb{T}_{in}^{i+1} \cap \mathbb{T}_{mem}^{i+1}}} = \left\{\phi_i^{i+1} \circ m_{i+1}\right\}|_{\substack{T \in \mathbb{T}_{mem}^i \,:\, T \in \mathbb{T}_{out}^i \cap \mathbb{T}_{mem}^i \wedge \\ \phi_i^{i+1} T \in \mathbb{T}_{in}^{i+1} \cap \mathbb{T}_{mem}^{i+1}}}$ // le variabili che restano in memoria non devono cambiare di indirizzo

2. $r_i|_{\substack{T \in \mathbb{T}_{reg}^i \,:\, \phi_i^{i+1} T \in \mathbb{T}_{reg}^{i+1} \cap \mathbb{T}_{in}^{i+1} \wedge \\ T \in \mathbb{T}_{reg}^i \cap \mathbb{T}_{out}^i}} = \{\phi_i^{i+1} \circ r_{i+1}\}|_{\substack{T \in \mathbb{T}_{reg}^i \,:\, \phi_i^{i+1} T \in \mathbb{T}_{reg}^{i+1} \cap \mathbb{T}_{in}^{i+1} \wedge \\ T \in \mathbb{T}_{reg}^i \cap \mathbb{T}_{out}^i}}$ // le variabili che restano nei registri non devono muoversi

3. $m_{i+1}|_{T \in \mathbb{T}_{mem}^{i+1} \cap \mathbb{T}_{in}^{i+1} \,:\, \phi_{i+1}^i(T) \in \mathbb{T}_{reg}^i \cap \mathbb{T}_{out}^i}$ is "optimal"

4. $m_{i+1}|_{T \in \mathbb{T}_{out}^{i+1} \cap \mathbb{T}_{mem}^{i+1}}$ is "optimal"

5. $r_{i+1}|_{T \in \mathbb{T}_{in}^{i+1} \cap \mathbb{T}_{reg}^{i+1} : \phi_{i+1}^i T \in \mathbb{T}_{mem}^i \cap \mathbb{T}_{out}^i}$

Memory allocation pseudocode

$m_1, r_1 \leftarrow$ Annealing, StatAlloc
for $i \in [2, |MM|]$:
/* Define Partitions for current Matrix Mult */
$\mathbb{T}^i_{in}, \mathbb{T}^i_{out} \leftarrow \mathcal{P}_{IO}(\mathbb{T}^i)$
$\mathbb{T}^i_{mem}, \mathbb{T}^i_{reg} \leftarrow \mathcal{P}_{MR}(\mathbb{T}^i)$
/* Retrieve Partitions for previous Matrix Mult */
$\mathbb{T}^{i-1}_{in}, \mathbb{T}^{i-1}_{out} \leftarrow \mathcal{P}_{IO}(\mathbb{T}^{i-1})$
$\mathbb{T}^{i-1}_{mem}, \mathbb{T}^{i-1}_{reg} \leftarrow \mathcal{P}_{MR}(\mathbb{T}^{i-1})$
/* Define mapping ϕ and ϕ^{-1} */
$\phi^i_{i-1} \leftarrow (\mathbb{T}^{i-1}_{out} \mapsto \mathbb{T}^i_{in})$
$\phi^{i-1}_i \leftarrow (\mathbb{T}^i_{in} \mapsto \mathbb{T}^{i-1}_{out})$
/* Define flows */
$flows^i_{i-1} \leftarrow \{\}$
/* Define r_i */
$FreeRegisters \leftarrow \mathbb{D}$
$r_i \leftarrow \{\}$

// if something some output stays in registers when becomes input we dont want to move it
for $t \in \left(\mathbb{T}^i_{in} \cap \mathbb{T}^i_{reg}\right)$
if $\phi^{i-1}_i(t) \in \left(\mathbb{T}^{i-1}_{out} \cap \mathbb{T}^{i-1}_{reg}\right)$ // corresponding output was in registers
$r_i.insert((t, r_{i-1}(t)))$
$FreeRegisters.remove(r_{i-1}(t))$
for $t \in \left(\mathbb{T}^i_{in} \cap \mathbb{T}^i_{reg}\right)$
if $\phi^{i-1}_i(t) \in \left(\mathbb{T}^{i-1}_{out} \cap \mathbb{T}^{i-1}_{mem}\right)$ // corresponding output was in memory
$r \leftarrow$ choose a register from $FreeRegisters$
$r_i.insert((t, r))$
$FreeRegisters.remove(r)$
$flows^i_{i-1}.push(Flow(m_{i-1}(\phi^{i-1}_i(t)) \mapsto r))$

/* Define m_i */
$m_i \leftarrow \{\}$
/* Define the list of "free" temporaries, in the sense that they have no constraint on memory position */
$UnconstrainedTemps \leftarrow \mathbb{T}^i_{mem}$
$ConstrainedTemps \leftarrow \{\}$

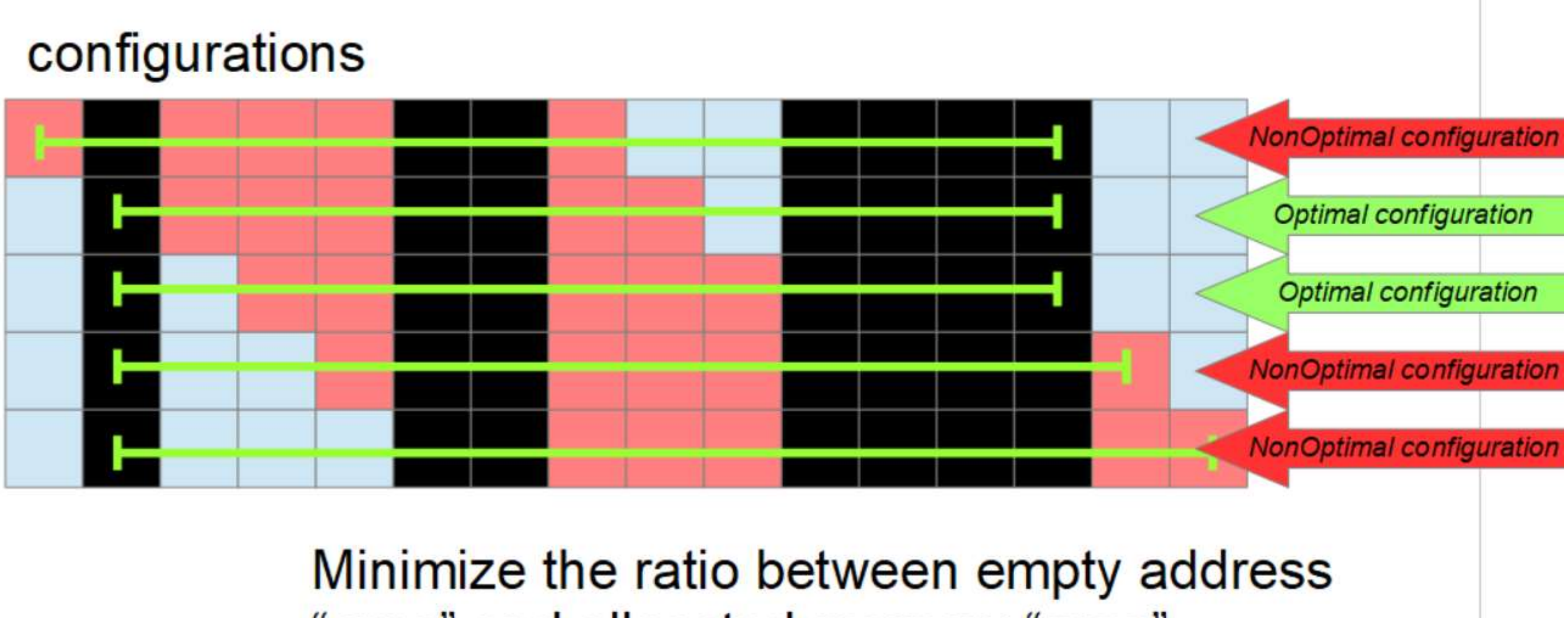
$(\star)$ /* Constraint the position in memory of what was previously in memory */
for $t \in \left(\mathbb{T}^i_{in} \cap \mathbb{T}^i_{mem}\right)$
if $\phi^{i-1}_i(t) \in \left(\mathbb{T}^{i-1}_{out} \cap \mathbb{T}^{i-1}_{mem}\right)$ // corresponding output was in memory
$m_i.insert((t, m_{i-1}(t)))$ // fix the constrained addresses
$UnconstrainedTemps.remove(t)$
$ConstrainedTemps.insert(t)$
/* Find $m_i(\mathbb{T}^i_{mem})$ image of the temps throu the map such that is "as dense as possible" */

$$m_i(\mathbb{T}^i_{mem}) \leftarrow MovingWindowDensityOptimization(\\ (ConstrainedTemps, \underbrace{m_i(ConstrainedTemps)}_{\text{We know these from } \star}), \\ UnconstrainedTemps \\ )$$

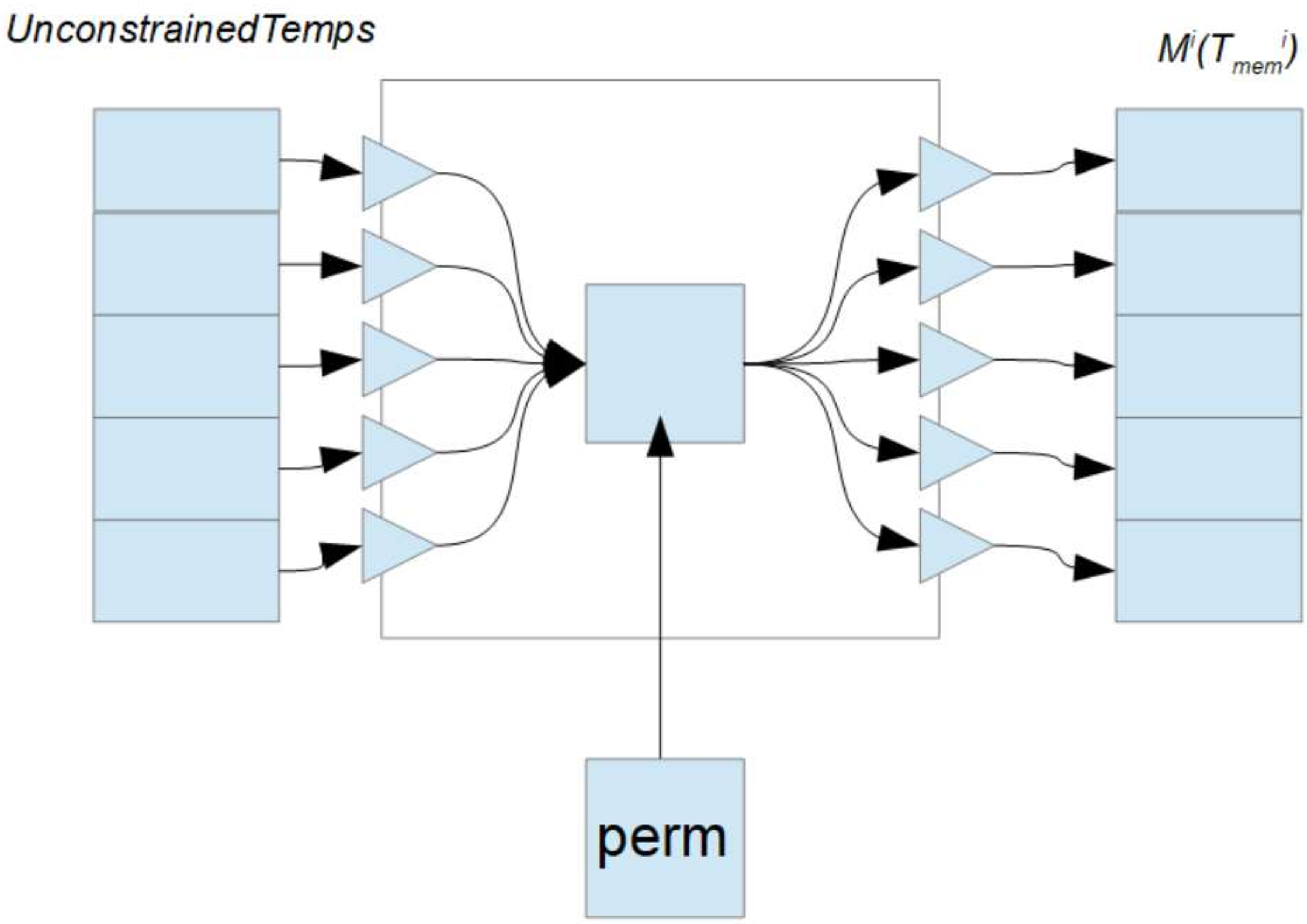
$\gamma(Permutation\ p) \leftarrow$ Returns the mapping between UnconstrainedTemps and $m^i(\mathbb{T}^i_{mem})$ induced by the permutation
$P \leftarrow$ permutation obtained thru annealing
for $u \in UnconstrainedTemps$
$m_i.insert((u, \gamma(P)(u)))$
/* Define the flows from registers in previous output to memory in current input */
for $t \in \left(\mathbb{T}^i_{in} \cap \mathbb{T}^i_{mem}\right)$
if $\phi^{i-1}_i(t) \in \left(\mathbb{T}^{i-1}_{out} \cap \mathbb{T}^{i-1}_{reg}\right)$
$flows^i_{i-1}.push(Flow(r_{i-1}(\phi^{i-1}_i(t)) \mapsto m_i(t)))$

MovingWindowDensityOptimization

Short graphical explanation



Permutation Optimization

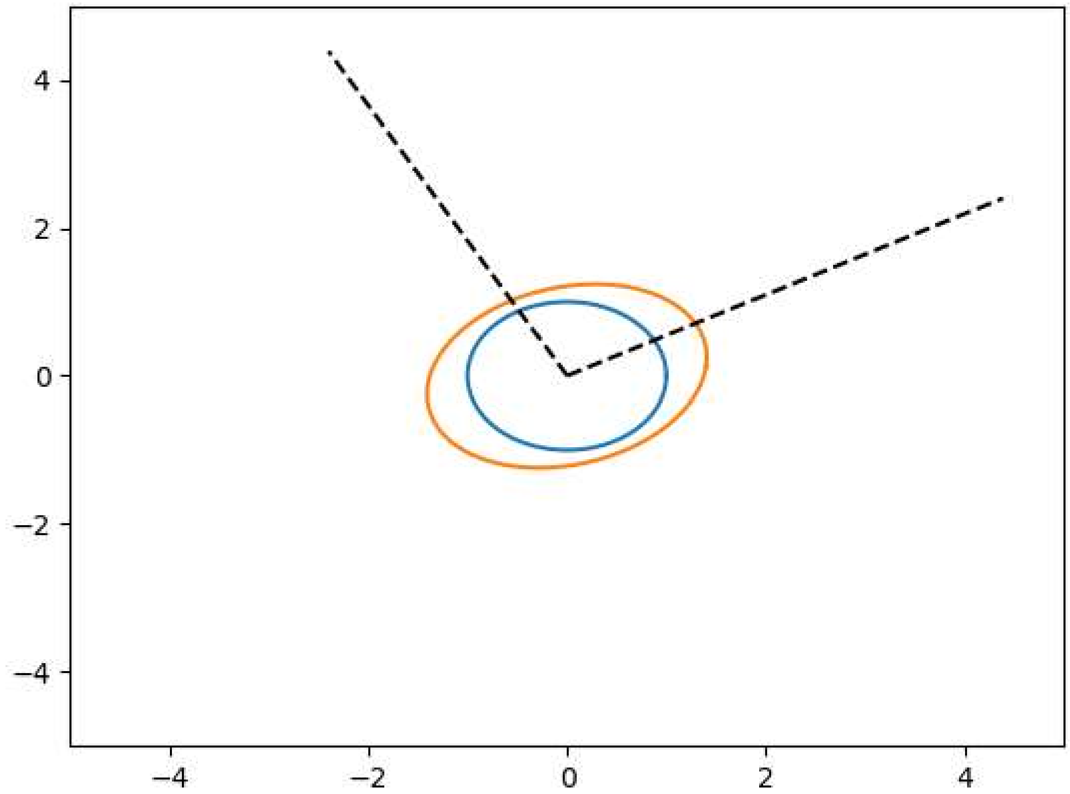


In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [43]:
```
theta = np.linspace(0 , 2 * np.pi, 100)
X = np.c_[ np.cos(theta), np.sin(theta)]
A  = np.random.randn(2,2)
A  = A.T @ A + np.eye(2) * 1.
AX = (A @ X.T).T
plt.plot(X[:,0],X[:,1])
plt.xlim(-5,5)
plt.ylim(-5,5)
plt.plot(AX[:,0],AX[:,1])

V = np.linalg.eig(A)[1]
V_points = np.array([
        (k * V.T).reshape(-1,4).flatten()
    for k in np.linspace(0,5)
])
plt.plot(V_points[:,0],V_points[:,1], '--', color = 'black')
plt.plot(V_points[:,2],V_points[:,3], '--', color = 'black')
```

Out[43]: [<matplotlib.lines.Line2D at 0x28570ee8610>]



In [ ]:
```

```