SPLAT: A Framework for Optimised GPU Code-Generation for SParse reguLar ATtention
Supplementary Material

AHAN GUPTA, University of Illinois at Urbana-Champaign, USA
YUEMING YUAN, University of Illinois at Urbana-Champaign, USA
DEVANSH JAIN, University of Illinois at Urbana-Champaign, USA
YUHAO GE, University of Illinois at Urbana-Champaign, USA
DAVID APONTE, Microsoft, USA
YANQI ZHOU, Google DeepMind, USA
CHARITH MENDIS, University of Illinois at Urbana-Champaign, USA

This document contains the supplementary material for SPLAT: A Framework for Optimised GPU Code-Generation for SParse reguLar ATtention. It contains extended ablations as well as proofs supporting the theorems in the main text.

A Extended Ablations

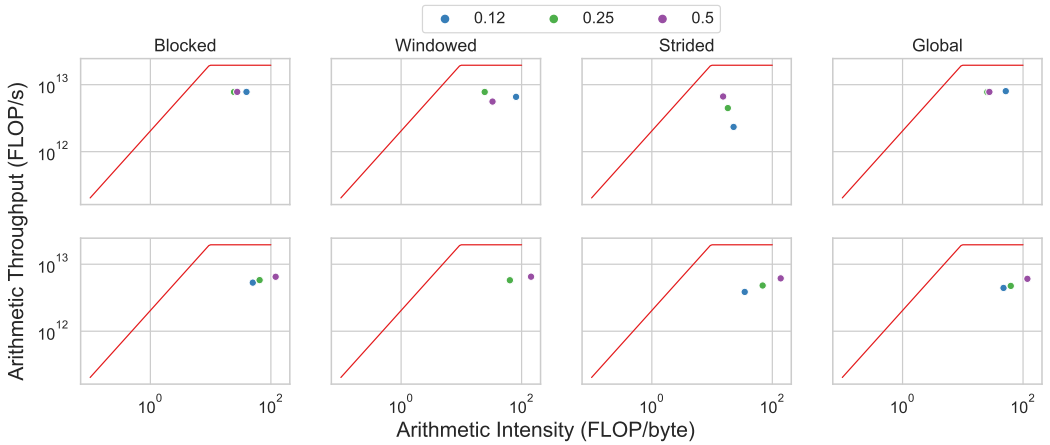

Fig. 1. Rooflines of SPLAT generated R-SDDMM and R-SpMM kernels at different moderate density levels for all the patterns. The top and bottom rows are the R-SDDMM and R-SpMM results respectively. For the windowed R-SpMM, the points corresponding to 0.12 and 0.25 density levels are overlapped.

Roofline analysis. We profile SPLAT generated R-SDDMM and R-SpMM kernels at moderate density levels: 12%, 25% and 50% for different patterns in figure 1 to identify how close to the

Authors' Contact Information: Ahan Gupta, University of Illinois at Urbana-Champaign, Champaign, USA, ag82@illinois.edu; Yueming Yuan, University of Illinois at Urbana-Champaign, Champaign, USA, yy28@illinois.edu; Devansh Jain, University of Illinois at Urbana-Champaign, Champaign, USA, devansh9@illinois.edu; Yuhao Ge, University of Illinois at Urbana-Champaign, Champaign, USA, yuhaoge2@illinois.edu; David Aponte, Microsoft, Seattle, USA, davidaponte@microsoft.com; Yanqi Zhou, Google DeepMind, Mountain View, USA, yanqiz@google.com; Charith Mendis, University of Illinois at Urbana-Champaign, Champaign, USA, charithm@illinois.edu.

theoretical peak throughput SPLAT generated kernels are. For the R-SDDMM kernel, we see an average of: 39% (blocked), 34% (windowed), 23% (strided), and 40% (global) of the theoretical peak across the density values. For the R-SpMM kernel, we see an average of: 30% (blocked), 31% (windowed), 25% (strided), and 26% (global) of the theoretical peak across the density values.

B Geometrical Definitions of Each Pattern

We define the geometry of the blocked, windowed, strided and global pattern. We do this by defining a function, f, on the point-set, P, of a regularly-sparse mask. Since P is regularly-sparse, every row in P is affine-compressible. The function f accordingly maps a row, x, in P to a 3-tuple consisting of the affine-indices - a, b - and number of non-zero values - nnz - that compress all the points that lie on x.

Definition 6. *Consider a regularly-sparse point-set, P. Let* $X = \{y_i | (x_i, y_i) \in P\}$ *with* $X(i) = \{y_j | (x_j, y_j) \in P \wedge y_j = i\}$. *We define* $f : X \rightarrow \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}$ *with the property that:*

$$\forall (x_j, y_j) \wedge (x_{j+1}, y_j) \in P \text{ such that, } y_j \in X(i), \frac{x_j - f(i).b}{f(i).a} + 1 = \frac{x_{j+1} - f(i).b}{f(i).a} \wedge |X(i)| = f(i).nnz$$

Where $(f(i).a, f(i).b, f(i).nnz) \in \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}$ *denotes the 3-tuple in the range of* f.

We can then define each pattern by constructing its relevant f function. For each of the definitions, N is the number of rows in the corresponding pattern ($|X|$ in definition 6).

Definition 7. *Define the blocked pattern, with block-width w as:*

$$\forall i \in [0, w], f(i) = (1, 0, w)$$
$$\forall i \in [w + 1, N - 1], f(i) = (1, (floor(i/w) - 1)w, 2w)$$

Definition 8. *Define the window pattern, with window-width w as:*

$$\forall i \in [0, w + 1], f(i) = (1, 0, w + 1 + i)$$
$$\forall i \in [w + 2, N - w - 2], f(i) = (1, i + w, 2w + 1)$$
$$\forall i \in [N - w - 1, N - 1], f(i) = (1, i + w, w + N - i)$$

Definition 9. *Define the strided pattern, with stride s as:*

$$\forall i \in [0, N - 1] \text{ such that, } i \pmod s = j, f(i) = (s, j, \lceil (N - j)/s \rceil)$$

Definition 10. *Define the global pattern, with number of global tokens g as:*

$$\forall i \in [0, g], f(i) = (1, 0, N)$$
$$\forall i \in [g + 1, N - 1], f(i) = (1, 0, g)$$

C Stretching in Polygonal Patterns

Theorem 2. *If we assume that the anchor of the thread block* TB_i *is present in a polygonal mask, i.e.* $Anc(TB_i) \in P$, *then* $|Cov(TB_i)|$ *would be maximum when its stretch factor* $Str(TB_i) = 1$.

Proof. The argument is pretty simple here.

If $|Cov(TB_i)| = mn$ for $|Str(TB_i)| = 1$, then it is already at its maximum since $|Cov(TB_i)| \leq |Comp(TB_i)| = mn$.

If $|Cov(TB_i)| < mn$ for $|Str(TB_i)| = 1$, then one of the right edge or the bottom edge is crossing the polygonal mask boundary. WLOG, let us assume that the right edge is outside the polygonal mask. If we were to increase the stretch of the thread block, we would only find more points not belonging to the mask. Thus, the $|Cov(TB_i)|$ will never increase beyond $|Str(TB_i)| = 1$. □

D Stretching in Strided Patterns

Theorem 3. *If we assume that the anchor of the thread block TB_i is present in a strided mask, i.e. $Anc(TB_i) \in P$ and that the stride X is divisible by its stretch factor $s = Str(TB_i)$, i.e. $X = \kappa s$, we can precisely calculate:*

$$|Cov(TB_i)| = \left\lceil \frac{m-n+1}{\kappa} \right\rceil \times n + \sum_{k=\lceil \frac{m-n+1}{\kappa} \rceil}^{\lceil \frac{m-1}{\kappa} \rceil} (m - k\kappa) + \sum_{k=1}^{\lceil \frac{n-1}{\kappa} \rceil} (n - k\kappa)$$

Proof. Let's assume that point p is present in both $Comp(TB_i)$ and P. Let $(x, y) = Anc(TB_i)$.
Since $(x, y) \in P$, $(x, y) = (\alpha_0 \kappa s + t_0, \beta_0 \kappa s + t_0)$ for some $\alpha_0, \beta_0 \in \mathbb{N}_0$, $t_0 \in \mathbb{Z}_{\kappa s}$
Since $p \in Comp(TB_i)$, $p = (x + is, y + js)$ for some $i \in \mathbb{Z}_n$, $j \in \mathbb{Z}_m$
Since $p \in P$, $p = (\alpha_1 \kappa s + t_1, \beta_1 \kappa s + t_1)$ for some $\alpha_1, \beta_1 \in \mathbb{N}_0$, $t_1 \in \mathbb{Z}_{\kappa s}$
Therefore, $x + is = \alpha_1 \kappa s + t_1$ and $y + js = \beta_1 \kappa s + t_1$.
$\implies x + is - \alpha_1 \kappa s = y + js - \beta_1 \kappa s$
$\implies \alpha_0 \kappa s + t_0 + is - \alpha_1 \kappa s = \beta_0 \kappa s + t_0 + js - \beta_1 \kappa s$
$\implies \alpha_0 \kappa s + is - \alpha_1 \kappa s = \beta_0 \kappa s + js - \beta_1 \kappa s$
$\implies \alpha_0 \kappa + i - \alpha_1 \kappa = \beta_0 \kappa + j - \beta_1 \kappa$ [Since $s \neq 0$]
$\implies \kappa(\alpha_0 - \alpha_1 - \beta_0 + \beta_1) = j - i$
$\implies \kappa$ divides $(j - i)$ [Since $\alpha_0, \alpha_1, \beta_0, \beta_1 \in \mathbb{N}_0$]

We know that $|Comp(TB_i)| = mn$
$|Cov(TB_i)|$ is the number of points present in both $Comp(TB_i)$ and P which is equivalent to number of solutions of $(i, j) \in \mathbb{Z}_n \times \mathbb{Z}_m$ satisfying κ divides $(j - i)$.

As shown in Fig 2, we can partition the thread block into three sections.
Green section contains threads with $(j - i) \in \{-n, \ldots, -1\}$.
Pink section contains threads with $(j - i) \in \{0, \ldots, m - n\}$.
Blue section contains threads with $(j - i) \in \{m - n + 1, \ldots, m\}$.

$$\text{Number of solutions in green section} = \sum_{k=1}^{\lceil \frac{n-1}{\kappa} \rceil} (n - k\kappa)$$

$$\text{Number of solutions in pink section} = \left\lceil \frac{m-n+1}{\kappa} \right\rceil \times n$$

$$\text{Number of solutions in blue section} = \sum_{k=\lceil \frac{m-n+1}{\kappa} \rceil}^{\lceil \frac{m-1}{\kappa} \rceil} (m - k\kappa)$$

Therefore,

$$|Cov(TB_i)| = \left\lceil \frac{m-n+1}{\kappa} \right\rceil \times n + \sum_{k=\lceil \frac{m-n+1}{\kappa} \rceil}^{\lceil \frac{m-1}{\kappa} \rceil} (m - k\kappa) + \sum_{k=1}^{\lceil \frac{n-1}{\kappa} \rceil} (n - k\kappa)$$

□

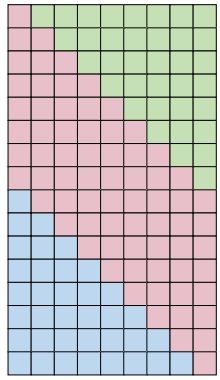

Fig. 2. Partitioning the Thread Block based on $j - i$ value

Theorem 4. *Stronger form: If we assume that the anchor of the thread block TB_i is present in a strided mask, i.e. $Anc(TB_i) \in P$ with stretch factor $s = Str(TB_i)$, we can precisely calculate: $|Cov(TB_i)| = f(\kappa)$ where*

$$f(\kappa) = \left\lfloor \frac{m-n+1}{\kappa} \right\rfloor \times n + \sum_{k=\lceil \frac{m-n+1}{\kappa} \rceil}^{\lceil \frac{m-1}{\kappa} \rceil} (m - k\kappa) + \sum_{k=1}^{\lceil \frac{n-1}{\kappa} \rceil} (n - k\kappa) \qquad \text{and} \qquad \kappa = \frac{X}{gcd(s, X)}$$

Proof. Similar to previous proof, we can conclude $(\alpha_0 - \alpha_1 - \beta_0 + \beta_1)d = (j - i)s$.
$\implies \frac{(j-i)s}{d} \in \mathbb{Z}$.
Let $\tau = gcd(s, X)$, i.e. $X = \kappa\tau$, $s = \eta\tau$, $gcd(\kappa, \eta) = 1$.
$\implies \frac{(j-i)\eta}{\kappa} \in \mathbb{Z}$.
$\implies \kappa$ divides $(j - i)$
The rest of the proof remains the same. □

Theorem 5. *Suppose we have a strided pattern with stride X. λ^s is the number of thread-blocks generated by poset tiling with stretch factor s. Then we have that λ^s decreases as* $\gcd(s, X)$ *increases.*

Proof. By definition of strided pattern and poset tiling algorithm, we have no overlapping thread blocks. Thus, we can effectively compute $\lambda^s = \frac{|P|}{f(\kappa)}$. All terms in $f(\kappa)$ are monotonically decreasing with κ. λ^s monotonically increases with $\kappa = \frac{X}{gcd(s,X)}$. Thus, λ^s decreases as $gcd(s, X)$ increases. □

For strided pattern, $Cost(TB_{poset}) = \lambda^s s$. Since, λ^s decreases as $gcd(s, X)$ increases. $Cost(TB_{poset})$ will achieve maximum value for some $s \in factors(X)$, thereby, reducing the search space of stretch factor.

E Proof of Theorem 1

In Theorem 1, we make claims on the optimality bounds of the Poset Tiling Algorithm. We begin by rigorizing the geometrical definitions for each of the windowed, blocked, strided and global patterns in B. Following which, we prove the bound for the windowed and blocked patterns in E.1, and the strided pattern in E.2.

E.1 Structured Polygonal Patterns

Theorem 6. *Suppose we have a strided pattern with stride X. λ^s is the number of thread-blocks generated by poset-tiling with stretch factor s. Then we have that λ^s only decreases when* $\gcd(s, X)$ *increases.*

We prove this by computing λ^s as a monotonically decreasing function of $gcd(s, X)$.

Definition 11. *A mask P is considered a structured dense polygon if a significantly large horizontal subsection of the mask consisting of rh rows ($N - rh \ll N$) can be partitioned as r repeating slabs of shape $h \times l$ recursively shifted by l' columns.*

$$P = \left\{ \left(\left\lfloor \frac{y}{h} \right\rfloor l' + t,\ y \right) \;\middle|\; t \in \mathbb{Z}_l,\ y \in \mathbb{Z}_{rh} \right\}$$

While the mathematical definition looks restrictive, several transformers have sampling masks that can be parameterized by this definition. For example, the block pattern had $h = l' = w, l = 2w$, the window pattern had $h = l' = 1, l = 2w + 1$ and the global pattern had $h = 1, l' = 0, l = g$.

Definition 12. *Let us consider a Naive Tiling algorithm that partitions the mask P into patches where a patch is a region of m consecutive rows. It then tiles these patches from left to right.*
We define $w^{(k)}$ as the width of the k^{th} patch. If a patch overlaps with t (≥ 1) slabs, its width is $l + (t-1)l'$.

Theorem 7. *Given a structured dense polygon mask P with parameters (r, l, h, l') such that $gcd(m, h) = \kappa$ (i.e., $m = \tau_m \kappa$, $h = \tau_h \kappa$) and $\tau_m = \alpha\tau_h + \beta$, $\beta \in \mathbb{Z}_{\tau_h}$, the Naive Tiling Algorithm will generate a TB of size λ_{naive}.*

$$\lambda_{naive} = \begin{cases} \left(\frac{r}{\tau_m}\right)\left(\tau_h \left\lceil \frac{l+(\alpha-1)l'}{n} \right\rceil\right) & \beta = 0 \text{ (i.e., } h|m) \\ \left(\frac{r}{\tau_m}\right)\left((\beta-1)\left\lceil \frac{l+(\alpha+1)l'}{n} \right\rceil + (\tau_h - \beta + 1)\left\lceil \frac{l+\alpha l'}{n} \right\rceil\right) & \beta \neq 0 \end{cases}$$

Proof. P (rh rows) can be partitioned into $\frac{r}{\tau_m}$ horizontal cross-sections of P with $lcm(m, h) = \tau_m \tau_h \kappa$ rows.
Each horizontal cross-section will be covered by τ_h patches (m rows) and τ_m slabs (h rows).
Therefore, $\lambda_{naive} = (\frac{r}{\tau_m})(\sum_{i=1}^{\tau_h} \lceil \frac{w^{(i)}}{n} \rceil)$.
If $\beta = 0$, i.e. $h|m$, we have $\kappa = h$, $\tau_m = \alpha$, $\tau_h = 1$. The cross-section is covered by 1 patch overlapping with τ_m slabs with width $(l + (\tau_m - 1)h)$ $[= (l + (\alpha - 1)l')]$. This proves the first case of the equation.
If $\beta \neq 0$ and $\alpha = 0$ (i.e., $m < h$), we have $\tau_m = \beta$. Every patch will overlap with either 1 slab or 2 slabs. Since there are τ_m slabs, exactly $(\tau_m - 1)$ $[= (\beta - 1)]$ patches will overlap with 2 slabs and will have width $(l + l')$ $[= (l + (\alpha + 1)l')]$. Rest of the $(\tau_h - \tau_m + 1)$ $[= (\tau_h - \beta + 1)]$ patches will overlap with 1 slab and will have width l $[= (l + \alpha l')]$. This proves the second case of the equation for $\alpha = 0$.
If $\beta \neq 0$ and $\alpha \neq 0$ (i.e., $m > h$), every patch will overlap with either $(\alpha + 1)$ slabs or $(\alpha + 2)$ slabs. Since there are τ_h patches, exactly $(\tau_h - 1)$ slabs will overlap with 2 patches.
Say x patches overlap with $(\alpha + 2)$ slabs. We get $(x)(\alpha + 2) + (\tau_h - x)(\alpha + 1) = \tau_m + (\tau_h - 1)$. Solving for x gives us $x = (\tau_m - \alpha\tau_h) - 1 = \beta - 1$.
There are $(\beta - 1)$ patches that will overlap with $(\alpha + 2)$ slabs and will have width $(l + (\alpha + 1)l')$. Rest of the $(\tau_h - \beta + 1)$ patches will overlap with $(\alpha + 1)$ slab and will have width $(l + \alpha l')$. This proves the second case of the equation for $\alpha \neq 0$. □

Theorem 8. *For a given structured dense polygon mask P with parameters (r, l, h, l'), Algorithm A and Tiling Algorithm generate TB_A and TB_{naive}. We define $\lambda^{(k)}$ and $\lambda^{(k)}_{naive}$ as the number of thread blocks with anchored in first k patches (km rows) for TB_A and TB_{naive}. $\delta^{(k)}$ thread blocks in TB_A are anchored in the first k patches and aren't contained within the first k patches, in other words, thread blocks which overflow.*
For any $k \geq 1$, $\lambda^{(k)} - \delta^{(k)} \geq \lambda^{(k)}_{naive} - \sum_{i=1}^{k} \lceil \frac{w^{(i)} - l}{n} \rceil$.

Proof. We will prove the theorem by induction.
We know that $\lambda^{(k)}_{naive} = \sum_{i=1}^{k} \lceil \frac{w^{(k)}}{n} \rceil$.

Base Case: We need to show that $\lambda^{(1)} - \delta^{(1)} \geq \lambda^{(1)}_{naive} - \lceil \frac{w^{(1)} - l}{n} \rceil$.
The minimum value of $\lambda^{(1)} - \delta^{(1)}$ is $\lceil \frac{l}{n} \rceil$ since the first row of the first patch needs to be covered by TB_A.

$$\begin{aligned}
\lambda^{(1)} - \delta^{(1)} &\geq \left\lceil \frac{l}{n} \right\rceil \\
&\geq \left\lceil \frac{w^{(1)}}{n} \right\rceil - \left\lceil \frac{w^{(1)} - l}{n} \right\rceil \quad \text{(Since, } \lceil x \rceil + \lceil y \rceil \geq \lceil x + y \rceil\text{)} \\
&= \lambda^{(1)}_{naive} - \left\lceil \frac{w^{(1)} - l}{n} \right\rceil
\end{aligned}$$

Hypothesis: Assuming $k \geq 1$, such that $\lambda^{(k)} - \delta^{(k)} \geq \lambda^{(k)}_{naive} - \sum_{i=1}^{k} \lceil \frac{w^{(i)} - l}{n} \rceil$.

Inductive Step: We need to show that $\lambda^{(k+1)} - \delta^{(k+1)} \geq \lambda^{(k+1)}_{naive} - \sum_{i=1}^{k+1} \lceil \frac{w^{(i)} - l}{n} \rceil$.
The minimum value of $\lambda^{(k+1)} - \delta^{(k+1)}$ is $\lambda^{(k)} + \lceil \frac{l - \delta^{(k)} n}{n} \rceil$ since the first row of the $(k+1)^{th}$ patch needs to be covered by TB_A. The only thread blocks that can cover this row are ones with the anchor in the first row of $(k+1)^{th}$ patch and the ones with the anchor in the k^{th} patch but overflow.

$$\begin{aligned}
\lambda^{(k+1)} - \delta^{(k+1)} &\geq \lambda^{(k)} + \left\lceil \frac{l - \delta^{(k)} n}{n} \right\rceil \\
&= \lambda^{(k)} - \delta^{(k)} + \left\lceil \frac{l}{n} \right\rceil \\
&\geq \lambda^{k}_{naive} - \sum_{i=1}^{k} \left\lceil \frac{w^{(i)} - l}{n} \right\rceil + \left\lceil \frac{l}{n} \right\rceil \quad \text{(Using the Hypothesis)} \\
&\geq \lambda^{k}_{naive} - \sum_{i=1}^{k} \lceil \frac{w^{(i)} - l}{n} \rceil + \left\lceil \frac{w^{(k+1)}}{n} \right\rceil - \left\lceil \frac{w^{(k+1)} - l}{n} \right\rceil \quad \text{(Since, } \lceil x \rceil + \lceil y \rceil \geq \lceil x + y \rceil\text{)} \\
&= \lambda^{(k)}_{naive} + \left\lceil \frac{w^{(k+1)}}{n} \right\rceil - \sum_{i=1}^{k+1} \left\lceil \frac{w^{(i)} - l}{n} \right\rceil \\
&= \lambda^{(k+1)}_{naive} - \sum_{i=1}^{k+1} \left\lceil \frac{w^{(i)} - l}{n} \right\rceil
\end{aligned}$$

□

Theorem 9. *For a given structured dense polygon mask P with parameters (r, l, h, l') such that $l' \approx h, l \gg m$, say the minimum number of thread blocks required to cover is λ_{opt}.*

$$\frac{\Delta\lambda}{\lambda_{opt}} = \frac{\lambda_{naive} - \lambda_{opt}}{\lambda_{opt}} \leq \frac{m}{l}$$

Proof. We know that $\lambda_{opt} \geq \frac{Nl}{mn}$.
From the previous theorem, we get that $\sum_{i=1}^{(N/m)} \lceil \frac{w^{(i)}-l}{n} \rceil \geq \lambda_{naive} - \lambda_{opt}$.
When $h < m$, we have $\sum_{i=1}^{(N/m)} \lceil \frac{w^{(i)}-l}{n} \rceil \leq \frac{N}{m} \frac{(\alpha+1)l'}{n}$.
Therefore, $\frac{\lambda_{naive}-\lambda_{opt}}{\lambda_{opt}} \leq \frac{(\alpha+1)l'}{l} \approx \frac{m}{l}$
When $h > m$, we have $(\tau_m - 1)$ patches that overlap 2 slabs for every τ_h patches. In other words, for every τ_h patches, there are $(\tau_m - 1)$ patches such that $w^{(i)} - l$ is l' while the rest of patches have $w^{(i)} - l$ as 0.
Therefore, $\sum_{i=1}^{(N/m)} \lceil \frac{w^{(i)}-l}{n} \rceil = \frac{N}{m} \frac{(\tau_m-1)l'}{\tau_h n} \leq \frac{N}{m} \frac{\tau_m l'}{\tau_h n} = \frac{Nl'}{hn}$.
Therefore, $\frac{\lambda_{naive}-\lambda_{opt}}{\lambda_{opt}} \leq \frac{l'm}{hl} \leq \frac{m}{l}$ □

Finally, we get $\frac{\lambda_{naive}}{\lambda_{opt}} \leq 1 + \frac{m}{l}$. We note that the Naive Tiling Algorithm is essentially the Poset Tiling Algorithm applied to each patch. Thus, we can argue that $\lambda_{poset} \leq \lambda_{naive}$.

This gives us $\frac{\lambda_{poset}}{\lambda_{opt}} \leq 1 + \frac{m}{l}$. As explained in §C, the optimal stretch factor for polygonal patterns is 1, so we get $\phi_{CMR} = 1$.

Thus, we prove that $\frac{Cost(TB_{poset})}{Cost(TB_{opt})} \leq 1 + \frac{m}{l}$.

E.2 Strided Patterns

By definition of strided pattern and poset tiling algorithm, we have no redundant compute (ϕ_R) and minimum thread-divergence (ϕ_{TD}). Thus, for a given stretch s, $\lambda_{opt} = \lambda_{poset}$. The stretch factor selection algorithm searches the space of the stretch factors and selects the stretch factor that minimizes our cost model. Thus, trivially, $Cost(TB_{opt}) = Cost(TB_{poset})$.

Finally, theorem 1 directly follows from the results of Sections E.1 & E.2.

Theorem 1. *Given point-set P and an arrangement of thread-blocks $TB_{poset} = \{TB_1, TB_2, ..., TB_{\lambda_{poset}}\}$, each of size $m \times n$, generated by algorithm 1 to cover P. Let $TB_{opt} = \{TB_1, TB_2, ..., TB_{\lambda_{opt}}\}$ be the arrangement of lowest possible cost to cover P. For the definitions of the windowed, blocked & strided patterns in B, we have for the windowed and blocked pattern that:*

$$\frac{Cost(TB_{poset})}{Cost(TB_{opt})} \leq 1 + \frac{m}{l} \tag{1}$$

Where l is the maximum number of points in a row of the mask. Moreover, for the strided pattern, the cost of the arrangement is optimal.