Supplementary Material for “A GPU Parallel Algorithm for Computing Morse-Smale Complexes”

Varshini Subhash, Karran Pandey and Vijay Natarajan, *Member, IEEE*

Abstract

This document presents additional statistics and experimental results supporting the paper “A GPU Parallel Algorithm for Computing Morse-Smale Complexes”. It includes a table that provides detailed statistics on the number of critical points and other quantities that help understand the available parallelism for the computation. Experimental results on the performance of the simplification algorithm for additional datasets, and results of the investigation of different configurations for grid subdivision and cancellation are presented. Finally, it presents GPU memory usage statistics for large datasets and pseudo code for different steps of the GPU algorithm.

1 MS COMPLEX COMPUTATION : PERFORMANCE ANALYSIS

Table 1 shows the number of junction nodes, 1-saddles, and 2-saddles for all datasets. Figure 1 shows the variation in matrix sparsity with an increase in dataset size (left to right). All input matrices - A_{1s-j}, B_{j-j}, B^{*}_{j-2s}, D_{1s-2s} and the final output matrix containing all 1-saddle–2-saddle connections are shown. Table 1 analyzes the number of sparse matrix multiplication (SpMM) operations between A_{1s-j} and B_{j-j}, which is the bottleneck in the gradient path counting algorithm.

Dataset	Size	Number of Critical Points	Junction Nodes	1-Saddles	2-Saddles	SpMM Iters
Silicium	98×34×34	1345	6595	527	530	29
Fuel	64×64×64	783	1883	304	266	56
Neghip	64×64×64	6193	17129	2963	1588	79
Tooth	103×94×161	827973	782493	319161	305313	30
Hydrogen	128×128×128	26725	48758	12834	7255	104
Shockwave	64×64×512	2477	5978	1063	898	230
Lobster	301×324×56	1201727	1463191	380903	491555	235
Ventricles	256×256×124	6073455	5032893	2447574	2135388	35
Engine	256×256×128	1541859	2369472	555341	589395	309
Bonsai	256×256×256	567133	1455152	167356	240688	495
Aneurysm	256×256×256	97319	164590	20678	44771	275
Foot	256×256×256	2387205	2716015	872346	913629	419
Turbulence	256×256×256	1474891	3496833	660342	484356	85
Skull	256×256×256	5786993	7275999	2249234	2090146	163
Angio	416×512×112	17811553	14615772	6988385	6470567	21
Isabel-Precip	500×500×100	1705641	1880804	614968	657219	730
Heptane	302×302×302	207431	593989	73659	78871	241

TABLE 1
Statistics for all datasets. Total number of critical points (including saddles and extrema), junction nodes, 1-saddles, and 2-saddles help understand the available parallelism for the saddle-saddle path computation. The number of $A_{1s-j} \times B_{j-j}$ sparse matrix multiplication (SpMM) iterations is indicative of the time required for the parallel gradient path counting step.

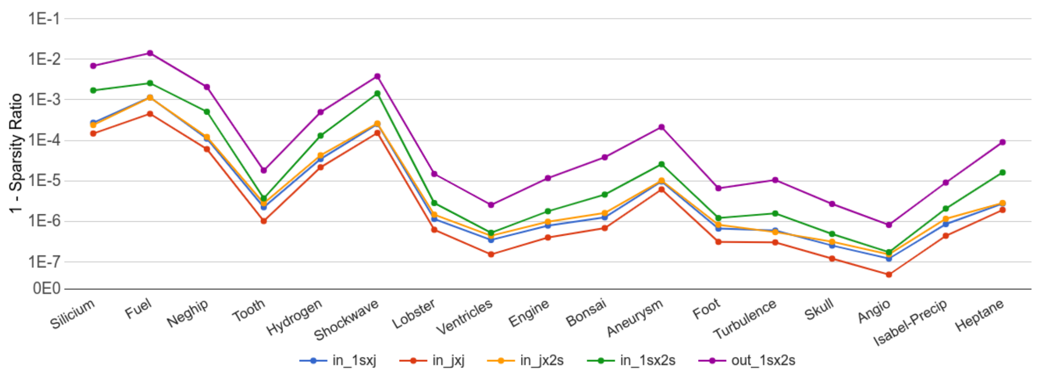

Fig. 1. Variation in matrix sparsities for the four input sparse matrices and the output matrix that captures the 1-saddle–2-saddle connections. Sparsity ratios of A_{1s-j} and B^{*}_{j-2s} are nearly identical. The matrix B_{j-j} exhibits highest sparsities. Smaller datasets have matrices with lower sparsity ratios, except Tooth which is noisy. In general, we notice that the matrices are highly sparse, with an overall increase in sparsity ratio with the size of the dataset. Cleaner datasets such as Aneurysm and Heptane display lower sparsity.

2 MS COMPLEX SIMPLIFICATION : PERFORMANCE ANALYSIS

We present additional details on the quality of simplification obtained by our GPU parallel algorithm for different datasets. We analyze the number of residual critical points, residual arcs, and strangulations below the chosen 5% threshold. Further, we study the distribution of critical points by index for all four configurations – the non-conservative and conservative approaches together with the two sequences of slicing for grid subdivision, XYZ + XYZ and XX + YY + ZZ.

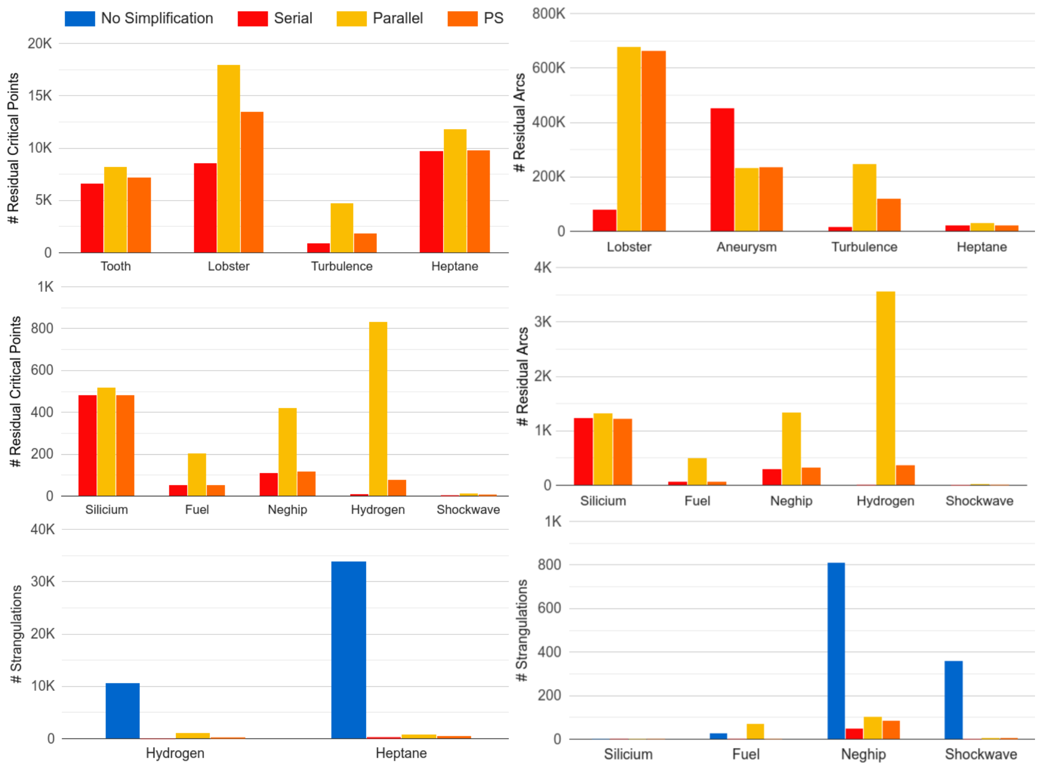

Fig. 2. Comparing the residual number of critical points, arcs, and strangulations (whose persistence is below the threshold) in the simplified MS complex between serial and parallel simplification (5% threshold). PS denotes the execution of six iterations of parallel simplification followed by serial simplification. (top to bottom) In small datasets, we notice close matches for the residual number of critical points with the exception of Hydrogen, Lobster and Turbulence due to excessive creation of strangulations. Parallel cancellation leads to a larger residual number, neutralized by the subsequent serial cancellation. Number of residual arcs is a close match in small datasets and is smaller in Aneurysm. An increase in Hydrogen, Lobster and Turbulence is again attributed to excessive strangulations. The number of strangulations are similar in almost all small datasets, and we observe a sharp reduction from the baseline count (pre-simplification).

Figure 2 depicts the residual critical points, residual arcs, and number of strangulations below a 5% threshold for our chosen configuration – non-conservative with XX+YY+ZZ grid subdivision. Figure 3 shows the number of strangulations below a 5% threshold for all four configurations, thus informing our final recommendation. Figure 4 depicts the distribution of residual critical points by index, *i.e.* minima, 1-saddles, 2-saddles, and maxima for all configurations. Table 4 shows GPU memory usage estimates for large data sizes.

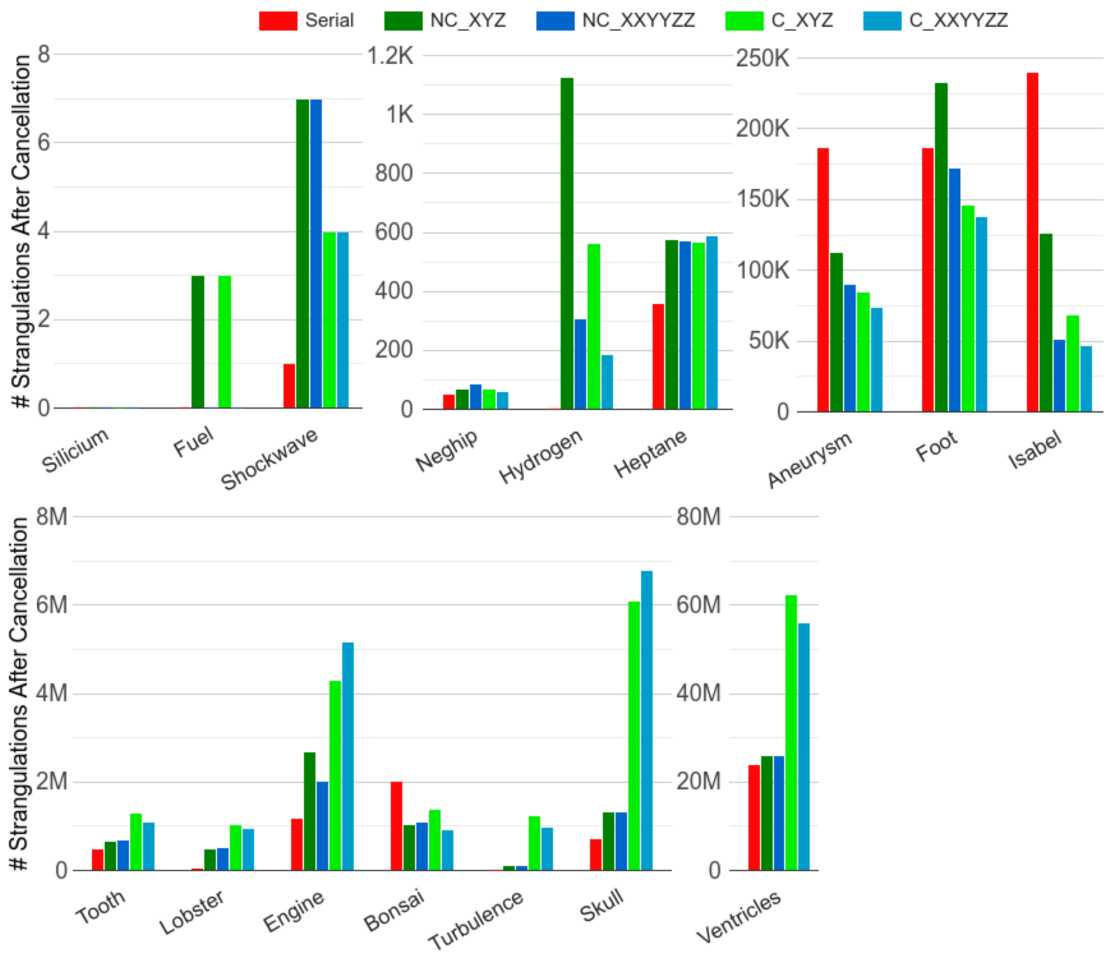

Fig. 3. Comparative plot of the number of strangulations after serial simplification and parallel simplification, for all four configurations – non-conservative (NC) and conservative (C) approaches with the two slicing orders, XYZ + XYZ and XX + YY + ZZ. We observe that NC_XYZ and NC_XXYYZZ perform better than their conservative counterparts for Tooth, Lobster, Engine, Turbulence, Skull and Ventricles. Small datasets such as Silicium, Fuel, Shockwave, Neghip, Heptane and larger ones like Aneurysm and Bonsai show similar results for all. Hydrogen, Foot and Isabel perform better with the NC_XXYYZZ and both conservative approaches. In case of Aneurysm, Isabel, and Bonsai, parallel simplification creates fewer strangulations when compared to the serial algorithm. Overall, we see consistently good performance with the NC_XXYYZZ configuration, which informs our final recommendation. We omit Angio due to the exponential increase in strangulations and long runtimes for the conservative cases. Raw results for Angio are reported in separate tables.

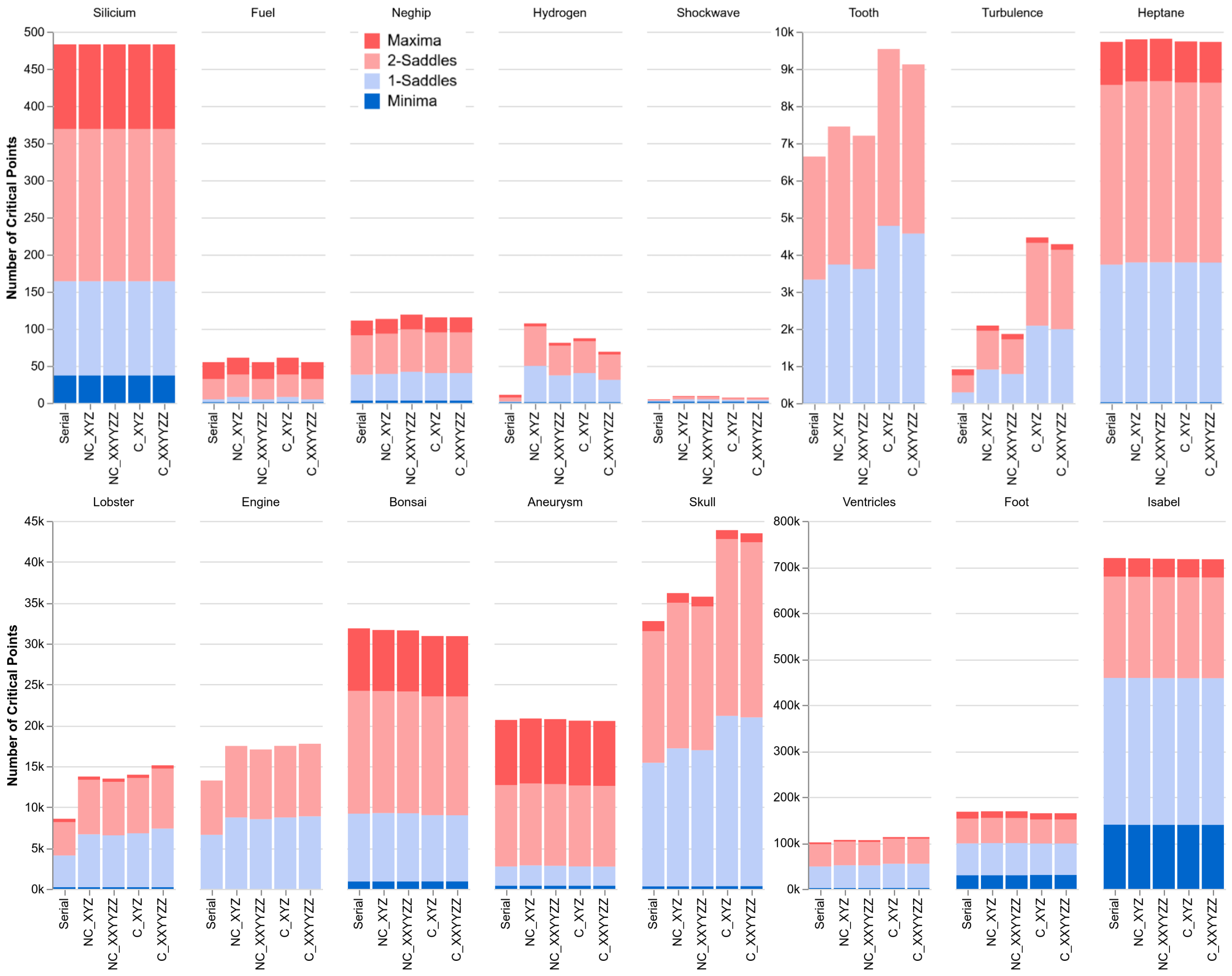

Fig. 4. Plot of the distribution of residual critical points after serial and parallel simplification. We observe nearly identical results for small datasets like Silicium, Fuel, Neghip, Shockwave and larger ones like Heptane, Bonsai, Aneurysm, Ventricles, Foot and Isabel. In Hydrogen, Tooth, Turbulence, Lobster, Engine and Skull, we note that the number of minima and maxima match well but the saddles show an increase with a corresponding increase in strangulations. We omit Angio in this figure due to the exponential increase in strangulations and large runtimes for the conservative cases. We include raw results for Angio in separate tables.

Simplification Method	# Minima	# 1-Saddles	# 2-Saddles	# Maxima	# Residual Critical Points	# Residual Arcs	# Strangulations
Serial	6619	120884	122019	7753	257275	85879985	56373884
NC_XYZ	6264	129514	130727	7476	273981	158197240	73886320
NC_XXYYZZ	6237	129622	130835	7449	274143	144489284	70355751

TABLE 2
Residual number of minima, 1-saddles, 2-saddles, maxima, total critical points, total arcs and strangulations below the chosen 5% threshold for Angio, after simplification. The non-conservative cases show a good match with the serial counterpart on all counts except for residual arcs which increases by an order of magnitude. The number of strangulations also shows an increase. We do not report the conservative cases, each of which take over 1 week (7 days) to run and are not practically usable. However, we do notice a sharp increase in strangulations below the threshold after subgrid simplification (Table 3) and the bulk of the runtime is spent on the serial simplification that follows.

	After Subgrid Simplification		
Simplification Method	# Critical Points	# Arcs	# Strangulations
NC_XYZ	301729	164307232	77600012
NC_XXYYZZ	294721	103771811	66012796
C_XYZ	345103	567206178	289709513
C_XXYYZZ	330883	511078497	262222528

TABLE 3
Number of critical points, arcs and strangulations below the chosen 5% threshold for Angio after subgrid simplification. For comparative purposes, we observe the following numbers before any simplification in Angio: critical points = 17,811,553, arcs = 63,681,723, strangulations = 3,561,462. The number of critical points and arcs behave similarly across both non-conservative and conservative cases. We observe an increase in strangulations in the conservative cases by an order of magnitude when compared to non-conservative cases, after subgrid simplification. We note that this could be a major contributing factor to the blow up in runtimes observed in the serial simplification that follows.

Dataset	Size	Utilized GPU Memory (GB)
Pancreas	$240\times512\times512$	9.9
Bunny	$512\times512\times361$	14.5
Present	$492\times492\times442$	16.6
Christmas Tree	$512\times499\times512$	20.5
Magnetic Reconnection	$512\times512\times512$	8.3
Zeiss	$680\times680\times680$	13.8

TABLE 4
GPU memory consumption estimates for 6 datasets that do not fit in memory and terminate prematurely when using gMSC. The GPU used in all experiments has a memory limit of 11 GB and usable memory between 8-11 GB. We present the GPU memory allocated up to the point of termination and note that each dataset has a different termination point depending on its size. In some cases, we also note a memory allocation larger than 11 GB because this includes all memory allocated until the point of termination, including memory that has been freed intermediately.

3 Algorithms

This section provides the pseudocode for the saddle reachability, DAG minor construction, and path counting algorithms for readers interested in the implementation details.

Algorithm 1: Saddle Reachability

```
Procedure: PARALLEL BFS
Input: 1-saddles
Output: Visited pairs
Init: $curr_{size}$, $next_{size}$, $current$, $next$
$curr_{\text{size}} \leftarrow num_{1\text{s}}$
$next_{\text{size}} \leftarrow 4 \times curr_{\text{size}}$
$current \leftarrow 1s$
while $true$ do
    for $(i = 0;\ i < curr_{size};\ i = i + 1)$ do
        $p \leftarrow pair\ of\ cofacet\ of\ current\ [i]$
        for $all\ valid\ pairs\ p$ do
            $next \leftarrow p$
            mark $p$ as visited
    $current.clear()$
    $current \leftarrow compaction\ of\ next$
    $curr_{\text{size}} = current.size()$
    if $curr_{size}$ == $0$ then
        break
    $next_{\text{size}} \leftarrow 4 \times curr_{\text{size}}$
    $next.resize\ (next_{\text{size}})$
```

Algorithm 2: DAG Minor Construction

```
Procedure: TRAVERSE MS GRAPH
Input: $input \leftarrow 1s,\ j$
Output: $dest_j,\ dest_{2s}$
Init: $srcs,\ dest_j,\ dest_{2s},\ paths$
$srcs \leftarrow input$
$dest_j.size(),\ dest_{2s}.size(),\ paths.size() \leftarrow 4 \times srcs_{\text{size}}$
while $true$ do
    for $(i = 0;\ i < srcs.size();\ i = i + 1)$ do
        $cfct \leftarrow cofacet\ of\ srcs$[i]
        if $cfct$ is a 2-saddle then
            $dest_{2s} \leftarrow cfct$
            return
        else
            $p \leftarrow pair\ of\ cfct$
            if $visited[p]$ && $junc[p]$ then
                $dest_j \leftarrow p$
            else if $visited[p]$ && ! $junc[p]$ then
                $paths \leftarrow p$
    $srcs.clear()$
    $srcs \leftarrow compaction\ of\ paths$
    if $srcs.size()$ == $0$ then
        break
```

Algorithm 3: Path Counting

```
Procedure: MATRIX MULTIPLY
Input: $A_{1s\times j}$, $B_{j\times j}$, ${B^*}_{j\times 2s}$, $D_{1s\times 2s}$
Output: Graph $G_{1s\times 2s}$
Init: $storage_{1s\times j}$, $C_{1s\times j}$, ${C^*}_{1s\times 2s}$
$storage_{1s\times j} \leftarrow A_{1s\times j}$
while $true$ do
    $C_{1s\times j} \leftarrow A_{1s\times j} \times B_{j\times j}$
    if $iter$ != 1 then
        $storage_{1s\times j} \leftarrow A_{1s\times j} + storage_{1s\times j}$
    if $C.size() == 0$ then
        break
    $A_{1s\times j} \leftarrow C_{1s\times j}$
    $C.clear()$
    $iter$++
${C^*}_{1s\times 2s} \leftarrow storage_{1s\times j} \times {B^*}_{j\times 2s}$
$G_{1s\times 2s} \leftarrow D_{1s\times 2s} + {C^*}_{1s\times 2s}$
```