

GRADES-NDA 2019



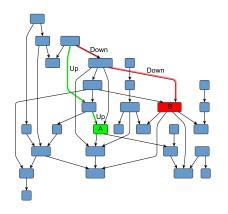
Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication

Nikita Mishin, Iaroslav Sokolov, Egor Spirin, Vladimir Kutuev, Egor Nemchinov, Sergey Gorbatyuk, **Semyon Grigorev**

> JetBrains Research, Programming Languages and Tools Lab Saint Petersburg University

> > June 30, 2019

Context-Free Path Querying



Navigation through a graph

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form Upⁿ Downⁿ?
- Find all paths of form
 Upⁿ Downⁿ which start from the node A

Context-Free Path Querying: Relational Query Semantics

- $\mathbb{G} = (\Sigma, N, P)$ context-free grammar in normal form
 - ▶ $A \rightarrow BC$, where $A, B, C \in N$
 - ▶ $A \rightarrow x$, where $A \in N, x \in \Sigma$
 - $L(\mathbb{G}, A) = \{ \omega \mid A \to^* \omega \}$
- G = (V, E, L) directed graph
 - $v \stackrel{l}{\rightarrow} u \in E$
 - $L \subset \Sigma$
- $\omega(\pi) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-2}} v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- $R_A = \{(n, m) \mid \exists n\pi m, \text{ such that } \omega(\pi) \in L(\mathbb{G}, A)\}$

[Scipy] Sparse matrices multiplication by using Scipy in Python programming language

- [Scipy] Sparse matrices multiplication by using Scipy in Python programming language
- [M4RI] Dense matrices multiplication by using m4ri2 library which implements the Method of Four Russians in C language

[GPU4R] Our own implementation of the Method of Four Russians in CUDA C.

- [GPU4R] Our own implementation of the Method of Four Russians in CUDA C.
- [GPU_N] Our own implementation of the naïve boolean matrix multiplication in CUDA C with boolean values treated as bits and packed into uint 32.

- [GPU4R] Our own implementation of the Method of Four Russians in CUDA C.
- [GPU_N] Our own implementation of the naïve boolean matrix multiplication in CUDA C with boolean values treated as bits and packed into uint _32.
- [GPU_Py] Manual implementation of naïve boolean matrix multiplication in Python by using numba compiler. Boolean values are packed into uint _32.

Evaluation: worst case

#V	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs
16	0.032	< 1	0.008	0.002	0.027	0.309
32	0.118	0.001	0.034	0.008	0.136	0.441
64	0.476	0.041	0.133	0.032	0.524	0.988
128	2.194	0.226	0.562	0.129	2.751	3.470
256	15.299	1.994	3.088	0.544	11.883	15.317
512	121.287	23.204	13.685	2.499	43.563	102.269
1024	1593.284	528.521	88.064	19.357	217.326	1122.055
2048	-	-	-	325.174	-	-

Evaluation: sparse

Graph	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs
G5k-0.001	10.352	0.647	0.113	0.041	0.216	5.729
G10k-0.001	37.286	2.395	0.435	0.215	1.331	35.937
G10k-0.01	97.607	1.455	0.273	0.138	0.763	47.525
G10k-0.1	601.182	1.050	0.223	0.114	0.859	395.393
G20k-0.001	150.774	11.025	1.842	1.274	6.180	-
G40k-0.001	-	97.841	11.663	8.393	37.821	-
G80k-0.001	-	1142.959	88.366	65.886	-	-

Example 2: Pseudoknot

RDF			Query G4							
Name	#V	#E	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs		
atm-prim	291	685	3	2	2	1	5	269		
biomed	341	711	3	5	2	1	5	283		
foaf	256	815	2	9	2	< 1	5	270		
funding	778	1480	4	7	4	1	5	279		
generations	129	351	3	3	2	< 1	5	273		
people pets	337	834	3	3	3	1	7	284		
pizza	671	2604	6	8	3	1	6	292		
skos	144	323	2	4	2	< 1	5	273		
travel	131	397	3	5	2	< 1	6	268		
unv-bnch	179	413	2	4	2	< 1	5	266		
wine	733	2450	7	6	4	1	7	294		
RI	RDF		Query G ₅							
					Q.	uery 05				
Name	#V	#E	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs		
Name atm-prim	#V 291	#E 685	Scipy 1	M4RI < 1			GPU_Py	CuSprs 267		
					GPU4R	GPU_N		•		
atm-prim	291	685	1	< 1	GPU4R	GPU_N < 1	2	267		
atm-prim biomed	291 341	685 711	1 4	< 1 < 1	GPU4R 1 1	GPU_N < 1 < 1	2 5	267 280		
atm-prim biomed foaf	291 341 256	685 711 815	1 4 1	< 1 < 1 < 1	GPU4R 1 1 1 3	GPU_N < 1 < 1 < 1 < 1	2 5 2 4 2	267 280 263		
atm-prim biomed foaf funding	291 341 256 778	685 711 815 1480	1 4 1 2	< 1 < 1 < 1 < 1	GPU4R 1 1 1 3	GPU_N < 1 < 1 < 1 < 1 < 1 < 1	2 5 2 4	267 280 263 274		
atm-prim biomed foaf funding generations	291 341 256 778 129	685 711 815 1480 351	1 4 1 2	< 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1	GPU4R 1 1 1 3	GPU_N < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1	2 5 2 4 2	267 280 263 274 263		
atm-prim biomed foaf funding generations people_pets	291 341 256 778 129 337	685 711 815 1480 351 834	1 4 1 2 1	< 1 < 1 < 1 < 1 < 1 < 1 < 1	GPU4R 1 1 1 3 1 1	GPU_N < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 <	2 5 2 4 2 3	267 280 263 274 263 277		
atm-prim biomed foaf funding generations people_pets pizza	291 341 256 778 129 337 671	685 711 815 1480 351 834 2604	1 4 1 2 1 1 2	< 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1	GPU4R 1 1 3 1 2	GPU N < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 <	2 5 2 4 2 3 5	267 280 263 274 263 277 278		
atm-prim biomed foaf funding generations people_pets pizza skos	291 341 256 778 129 337 671 144	685 711 815 1480 351 834 2604 323	1 4 1 2 1 1 2 < 1	< 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1	GPU4R 1 1 1 3 1 1 2 1	GPU N < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 < 1 <	2 5 2 4 2 3 5	267 280 263 274 263 277 278 265		

Example 2: Pseudoknot

RDF			Query G ₅						
Name	#V	#E	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs	
atm-prim	291	685	1	< 1	1	< 1	2	267	
biomed	341	711	4	< 1	1	< 1	5	280	
foaf	256	815	1	< 1	1	< 1	2	263	
funding	778	1480	2	< 1	3	< 1	4	274	
generations	129	351	1	< 1	1	< 1	2	263	
people pets	337	834	1	< 1	1	< 1	3	277	
pizza	671	2604	2	< 1	2	< 1	5	278	
skos	144	323	< 1	< 1	1	< 1	2	265	
travel	131	397	1	< 1	1	< 1	3	271	
unv-bnch	179	413	1	< 1	1	< 1	3	266	
wine	733	2450	1	< 1	3	< 1	3	281	

Contact Information

- Semyon Grigorev:
 - s.v.grigoriev@spbu.ru
 - Semen.Grigorev@jetbrains.com
- Polina Lunina:
 - ▶ lunina polina@mail.ru
- Trained models: https://github.com/YaccConstructor/YC.Bio

Thanks!