Krzysztof Nowicki

e-mail	knowicki@cs.uni.wroc.pl
www	`https://sites.google.com/cs.uni.wroc.pl/knowicki`

Wykształcenie

10.2015–09.2020	studia doktoranckie na Uniwersytecie Wrocławskim, na kierunku Informatyka promotor: prof. dr hab. Tomasz Jurdziński
09.2015	tytuł magistra informatyki temat pracy magisterskiej: *Multiple selection in MapReduce* promotor: prof. dr hab. Tomasz Jurdziński
10.2013–09.2015	studia magisterskie na Uniwersytecie Wrocławskim, na kierunku Informatyka
10.2009–09.2013	studia licencjackie na Uniwersytecie Wrocławskim, na kierunku Informatyka

Dłuższe wizyty naukowe

od 01.11.2019 do 30.04.2020	Staż doktorski (w ramach grantu z programu Etiuda7) w ETH Zurich opiekun naukowy: prof. Mohsen Ghaffari
od 24.06.2019 do 05.09.2019	Staż w IBM Research opiekun naukowy: Krzysztof Onak

Granty

10.2019–09.2020	*Algorytmy grafowe w różnych modelach obliczeń rozproszonych i równoległych.* Projekt sfinansowany przez Narodowe Centrum Nauki w ramach konkursu ETIUDA 7 (projekt nr 2019/32/T/ST6/00566)

Wyróżnienia / nagrody

05.2020	Stypendium START, przyznawane wybitnym młodym uczonym ze znaczącymi sukcesami w swojej dziedzinie nauki, przez Fundację na rzecz Nauki Polskiej
11.2018	Stypendium im. Hugona Steinhausa, przyznawane doktorantom przez miasto Wrocław, za osiągnięcia w zakresie nauk matematycznych

Referaty zaproszone

Discrete Algorithms at CanaDAM 2019	MST in O(1) rounds of Congested Clique
HALG 2018	MST in O(1) rounds of Congested Clique

Krótko o zainteresowaniach naukowych

W mojej pracy naukowej głównie zajmuję się algorytmami dla modeli obliczeń równoległych i rozproszonych. W szczególności interesują mnie modele, w których dane wejściowe problemu podzielone są pomiędzy pewną liczę maszyn, które wykonują obliczenia w synchronicznych rundach. Każda runda składa się z fazy lokalnych obliczeń i fazy komunikacji. Główną miarą złożoności algorytmów działających w takich modelach jest liczba rund potrzebna do rozwiązania rozważanego problemu algorytmicznego. W moich badaniach skupiam się głównie na dwóch modelach obliczeń, które realizują mniej więcej ten scenariusz – MPC i Congested Clique [1, 2, 3, 4, 5, 6]. Niemniej, niektóre z moich wyników mają aplikacje w innych modelach obliczeń [7, 8, 9, 10, 3].

Kilka słów o modelach obliczeń

Model MPC [21] jest teoretycznym modelem który stara się opisywać frameworki służące do równoległego przetwarzania dużych zbiorów danych, takie jak MapReduce [12], czy Spark [25]. W modelu MPC dane wejściowe rozmiaru N są podzielone równomiernie pomiędzy pewną liczbę maszyn M, każda otrzymuje fragment wejścia rozmiaru co najwyżej S. Parametry są dobrane w taki sposób żeby $M * S = O(N)$.

Obliczenia są wykonywane w synchronicznych rundach, każda składa się z fazy lokalnych obliczeń oraz fazy komunikacji. W fazie komunikacji, każda para maszyn może wymienić ze sobą wiadomości, a jedyny warunek jest taki, żeby liczba wiadomości wysyłanych / odbieranych przez każdą maszynę w ciągu jednej fazy komunikacji nie przekroczyła S. Aby skrócić notację, będę się odnosił do modelu MPC z parametrem $S \in f(n)$ jako MPC($f(n)$).

Na początku moich badań, głównie zajmowałem się problemami grafowymi (dla n-wierzchołkowych grafów) w modelu Congested Clique, który jest bardzo podobny do MPC($\mathcal{O}(n)$). W szczególności algorytmy z [1, 6, 2], mimo że opowiedziane w języku Congested Clique, dają się częściowo [2, 4] lub w całości [1, 6] zaimplementować w MPC($\mathcal{O}(n)$). W późniejszym czasie, poszerzyłem zakres moich zainteresowań o algorytmikę w modelu MPC($\mathcal{O}(n^\alpha)$), gdzie α jest stałą mniejszą niż 1 [4, 5].

Model Congested Clique [24, 23] jest modelem obliczeń w którym rozważam głównie problemy grafowe – mamy w nim n procesorów, ponumerowanych od 1 do n i każdy procesor odpowiada wierzchołkowi wejściowego grafu. Na początku każdy procesor zna wszystkie krawędzie incydentne do odpowiadającego mu wierzchołka. Obliczenia są wykonywane w synchronicznych rundach, każda składa się z fazy lokalnych obliczeń oraz fazy komunikacji. W fazie komunikacji, każda para maszyn może wymienić ze sobą jedną wiadomość składającą się z $O(\log n)$ bitów.

Okazuje się, że ten model jest w stanie symulować wariant MPC($\mathcal{O}(n)$) [22] oraz pewne algorytmy z modelu Congested Clique mają aplikację w MPC($\mathcal{O}(n)$) [17, 11].

Moje najistotniejsze wyniki dotyczą:

- problemu minimalnego drzewa spinającego [1, 6],
- problemu minimalnego cięcia w grafie [2, 3, 4],
- algorytmów równoległych dla dynamicznych grafów [5].

Minimalne drzewa spinające: Badania nad modelem Congested Clique zostały zainicjowane w pracy[24, 23], gdzie autorzy pokazują $\mathcal{O}(\log \log n)$ rundowy deterministyczny algorytm dla problemu minimalnego drzewa spinającego. Przed 2018 rokiem wynik był poprawiany dwukrotnie: w [16] autorzy proponują $\mathcal{O}(\log \log \log n)$ rundowy algorytm, a w [14] autorzy proponują $\mathcal{O}(\log^* n)$ rundowy algorytm. Nasz wynik opisany w [1] ostatecznie ustala, że minimalne drzewo spinające da się obliczyć w $\mathcal{O}(1)$ rundach. Niemniej, wszystkie wymienione $o(\log \log n)$ rundowe algorytmy korzystają ze zrandomizowanych (i dość skomplikowanych) technik. W niedawnej pracy [6] pokazuję, że minimalne drzewo spinające daje się obliczyć w $\mathcal{O}(1)$ rundach, bez korzystania z randomizacji. Co więcej, algorytm z [6] wydaje się być *istotnie* prostszy niż wcześniejsze $o(\log \log n)$ rundowe algorytmy. Wszystkie wymienione algorytmy dla problemu minimalnego drzewa spinającego są sformułowane jako algorytmy dla modelu Congested Clique, ale mają również zastosowanie w modelu MPC($\mathcal{O}(n)$).

Minimalne cięcia: Głównym wynikiem zaprezentowanym w pracy[2] jest $\mathcal{O}(1)$ rundowy algorytm obliczający $(1 + o(1))$-aproksymację minimalnego cięcia. Wynik ten poprawiam w dwóch późniejszych pracach.

W pracy[3] proponujemy proponujemy nowy rodzaj losowych kontrakcji dla grafów prostych[1]. Nasza technika pozwala znacząco zmniejszyć rozmiar wejściowego grafu, zachowując przy tym nietrywialne[2] minimalne cięcia. Główną zaletą naszej techniki jest jej prostota, która umożliwia zastosowania w wielu modelach obliczeń. Jej aplikacje pozwoliły uzyskać:

- $\mathcal{O}(1)$ rundowy algorytmy w MPC i Congested Clique,
- sekwencyjny algorytm o złożoności: $\min\Big(\mathcal{O}(m\log n), \mathcal{O}(m + n\log^2 n)\Big)$,
- rozproszony algorytm (w modelu Congest) potrzebujący $\tilde{O}(n^{0.8}D^{0.2} + n^{0.9})$ rund[3].

W [4] pokazujemy, że problem znajdowania minimalnych cięć w grafach ważonych może być rozwiązany w $\mathcal{O}(1)$ rundach, w modelu MPC($\mathcal{O}(n)$), korzystając z adaptacji algorytmu Kargera [20]. Algorytm ten nie ma bezpośredniej aplikacji w modelu Congested Clique, niemniej pozwala znaleźć $(1 + o(1))$-aproksymację (ze współczynnikiem lepszym niż algorytm z pracy[2]) minimalnego cięcia w $\mathcal{O}(1)$ rundach. Ponadto, w pracy[4] proponujemy także zmodyfikowaną wersję innego algorytmu Kargera [19], która znajduje $(2 + \varepsilon)$-aproksymację minimalnego cięcia w ważonych grafach, w modelu MPC($\mathcal{O}(n^\alpha)$), dla stałej $\alpha < 1$, w $\mathcal{O}(\log n \log\log n)$ rundach.

Dynamiczne grafy: W [5] rozważamy algorytmy dla dynamicznych grafów, w modelu MPC($\mathcal{O}(n^\alpha)$), dla stałej $\alpha < 1$. Zakładamy, że dane wejściowe to dynamiczny graf G, który może być reprezentowany jako sekwencja grafów $G_1, G_2, \ldots$, taka że G_1 jest znany na początku działania algorytmu, a G_i powstaje z G_{i-1} poprzez k modyfikacji. Pojedyncza modyfikacja to albo usunięcie, albo dodanie krawędzi do grafu.

Celem jest utrzymywanie wyniku dla dynamicznego grafu, czyli zaprojektowanie struktury danych, która *utrzymuje* wynik dla sekwencji $G_1, G_2, \ldots$. W naszej pracy zakładamy, że struktura danych dla ciągu k modyfikacji na grafie G_i powinna zwrócić ciąg modyfikacji które trzeba zaaplikować do wyniku dla G_i, aby otrzymać wynik dla grafu G_{i+1}. Pokazujemy algorytmy dla 3 problemów: utrzymywania minimalnego drzewa spinającego [$\mathcal{O}(1)$ rund dla $k \in \mathcal{O}(S)$], utrzymywania 2-krawędziowo-spójnych składowych [$\mathcal{O}(1)$ rund dla $k \in \mathcal{O}(S)$] i utrzymywania maksymalnego (w sense zawierania) skojarzenia [$\mathcal{O}(\log\log n)$ rund dla $k \in \mathcal{O}(S)$]. Jednym z kluczowych elementów naszych algorytmów dla utrzymywania minimalnego drzewa spinającego i utrzymywania 2-krawędziowo-spójnych jest adaptacja struktury danych *top tree* do modelu MPC, która potencjalnie może mieć wiele innych zastosowań.

Badania na temat algorytmów dla dynamicznych grafów w MPC posługują się modelem, w którym mamy co najmniej dwa parametry i kilka możliwych miar złożoności algorytmu. Poza wymienionymi powyżej parametrami S, k i złożonością rundową, rozważane są jeszcze złożoność komunikacyjna, liczba aktywnych maszyn oraz praca wykonywana przez maszyny. Badania zostały zainicjowane dość niedawno [18] i pomimo kilku prac na ten temat [18, 13, 15], nie jest jeszcze oczywiste jak wyglądają (ani jak powinny wyglądać) relacje pomiędzy poszczególnymi parametrami a złożonością algorytmu. W związku z tym nasza praca[5] nie tylko przedstawia pewne wyniki algorytmiczne, ale również wnosi wkład do dyskusji na temat tego co powinno być celem w trakcie projektowania algorytmów dla dynamicznych grafów w modelu MPC.

[1] Grafy proste to grafy nieskierowane, nieważone grafy bez krawędzi-pętli i równoległych krawędzi

[2] Za trywialne cięcia uważamy te, które odcinają dokładnie jeden wierzchołek od reszty grafu.

[3] D jest średnicą grafu, a $\tilde{\mathcal{O}}$ oznacza, że w wyrażeniu opisującym złożoność występują dodatkowe polilogarytmiczne czynniki.

Moje publikacje

[1] Tomasz Jurdzinski and Krzysztof Nowicki. MST in $\mathcal{O}(1)$ Rounds of Congested Clique. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2620–2632, 2018.

[2] Mohsen Ghaffari and Krzysztof Nowicki. Congested Clique Algorithms for the Minimum Cut Problem. In *Proceedings of the Symposium on Principles of Distributed Computing, PODC*, pages 357–366, 2018.

[3] Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster Algorithms for Edge Connectivity via Random 2-Out Contractions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2020.

[4] Mohsen Ghaffari and Krzysztof Nowicki. Massively Parallel Algorithms for Minimum Cut. In *Proceedings of the Symposium on Principles of Distributed Computing, PODC*, PODC '20, page 119–128, New York, NY, USA, 2020. Association for Computing Machinery.

[5] Krzysztof Nowicki and Krzysztof Onak. Dynamic Graph Algorithms with Batch Updates in the Massively Parallel Computation Model. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, page not available yet, 2021.

[6] Krzysztof Nowicki. A Deterministic Algorithm for the MST Problem in Constant Rounds of Congested Clique. *CoRR*, abs/1912.04239, 2019.

[7] Tomasz Jurdzínski and Krzysztof Nowicki. Brief Announcement: On Connectivity in the Broadcast Congested Clique. In *Proceedings of the International Symposium on Distributed Computing, DISC*, pages 54:1–54:4, 2017.

[8] Tomasz Jurdzinski and Krzysztof Nowicki. On Range and Edge Capacity in the Congested Clique. In *Proceedings of the Theory and Practice of Computer Science - International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM*, pages 305–318, 2018.

[9] Tomasz Jurdzinski and Krzysztof Nowicki. Connectivity and Minimum Cut Approximation in the Broadcast Congested Clique. In *Structural Information and Communication Complexity - International Colloquium, SIROCCO*, pages 331–344, 2018.

[10] Tomasz Jurdzinski, Krzysztof Lorys, and Krzysztof Nowicki. Communication Complexity in Vertex Partition Whiteboard Model. In *Structural Information and Communication Complexity - International Colloquium, SIROCCO*, pages 264–279, 2018.

Bibliografia

[11] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Brief Announcement: Semi-MapReduce Meets Congested Clique. *CoRR*, abs/1802.10297, 2018.

[12] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*, pages 137–150, 2004.

[13] Laxman Dhulipala, David Durfee, Janardhan Kulkarni, Richard Peng, Saurabh Sawlani, and Xiaorui Sun. Parallel batch-dynamic graphs: Algorithms and lower bounds. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, page 1300–1319, 2020.

[14] Mohsen Ghaffari and Merav Parter. MST in Log-Star Rounds of Congested Clique. In *Proceedings of the Symposium on Principles of Distributed Computing, PODC*, pages 19–28, 2016.

[15] Seth Gilbert and Lawrence Li Er Lu. How fast can you update your mst? In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures, SPAA*, page 531–533, 2020.

[16] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward Optimal Bounds in the Congested Clique: Graph Connectivity and MST. In *Proceedings of the Symposium on Principles of Distributed Computing, PODC*, PODC '15, pages 91–100, 2015.

[17] James W. Hegeman and Sriram V. Pemmaraju. Lessons from the Congested Clique Applied to MapReduce. In *Structural Information and Communication Complexity - International Colloquium, SIROCCO*, pages 149–164, 2014.

[18] Giuseppe F. Italiano, Silvio Lattanzi, Vahab S. Mirrokni, and Nikos Parotsidis. Dynamic algorithms for the massively parallel computation model. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures, SPAA*, page 49–58, 2019.

[19] David R. Karger. Using randomized sparsification to approximate minimum cuts. In Daniel Dominic Sleator, editor, *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 424–432, 1994.

[20] David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, January 2000.

[21] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A Model of Computation for MapReduce. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 938–948, 2010.

[22] Christoph Lenzen. Optimal Deterministic Routing and Sorting on the Congested Clique. In *Proceedings of the Symposium on Principles of Distributed Computing, PODC*, pages 42–50, 2013.

[23] Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-Weight Spanning Tree Construction in $\mathcal{O}(\log \log n)$ Communication Rounds. *SIAM J. Comput.*, 35(1):120–131, 2005.

[24] Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. MST construction in $\mathrm{O}(\log \log n)$ communication rounds. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures, SPAA*, pages 94–100. ACM, 2003.

[25] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, 2010.