Paweł Gawrychowski

Rok urodzenia	1983
e-mail	gawry@cs.uni.wroc.pl
www	http://www.ii.uni.wroc.pl/~gawry

Przebieg kariery zawodowej	**2011–2013** postdoc w Max-Planck-Institut für Informatik w Saarbrücken **od 2012** adiunkt w Instytucie Informatyki Uniwersytetu Wrocławskiego (obecnie na urlopie naukowym) **2011–2012** asystent w Instytucie Informatyki Uniwersytetu Wrocławskiego (na urlopie naukowym od listopada 2011) **2008–2011** programista C^{++} w nasza-klasa.pl
Wykształcenie	**2007–2011** studia doktoranckie na Wydziale Matematyki i Informatyki Uniwersytetu Wrocławskiego, rozprawa *Pattern matching in compressed text* obroniona z wyróżnieniem 25.10.2011 **2002–2007** studia magisterskie na kierunku informatyka na Uniwersytecie Wrocławskim **1998–2002** XIV Liceum Ogólnokształcące we Wrocławiu
Wyróżnienia	**2013** Nagroda im. Witolda Lipskiego **2012** Best Student Paper Award na konferencji CPM 2012 **2011** Best Student Paper Award na konferencji ESA 2011 **2011** Best Paper Award na konferencji CIAA 2011 **2011** Stypendium START fundacji na rzecz Nauki Polskiej **2010** Stypendium im. Maxa Borna **2009** Best Student Paper Award na konferencjach MFCS 2009 i IWOCA 2009 **2004–2007** Stypendium za wyniki w nauce Ministerstwa Edukacji Narodowej i Sportu

Opis dotychczasowej pracy naukowej

W dotychczasowej pracy zajmowałem się głównie analizą i konstruowaniem efektywnych algorytmów dla problemów związanych z przetwarzaniem skompresowanych tekstów, gdzie efektywne oznacza o złożoności jak najbliższej liniowej.

Wyszukiwanie wzorca w skompresowanym tekście. W swojej rozprawie doktorskiej rozważałem problem wyszukiwania wzorca w tekstach skompresowanych metodami opartymi na schemacie Lempela-Ziva (a dokładniej, LZ/LZ77 i LZ78/LZW), uzyskując szereg wyników (podsumowanych w poniższej tabeli) poprawiających większość wcześniejszych rezultatów w tej dziedzinie.

LZW	$\mathcal{O}(n+M)$	SODA 2011
	$\mathcal{O}(n+m)$	w pełni skompresowane, STACS 2012
	$\mathcal{O}(n+M^{1+\epsilon})$	wiele wzorców, CPM 2012
	$\mathcal{O}(n \log M + M)$	
LZ	$\mathcal{O}(n \log \frac{N}{n} + M)$	ESA 2011

W pracy z konferencji SODA 2011 udowodniłem, że możliwe jest bardzo szybkie wyszukiwanie wzorca w tekście skompresowanym metodą LZW, poprawiając rozwiązania działające w czasie $\mathcal{O}(n \log M + M)$ lub $\mathcal{O}(n + M^2)$ podane przez Amira, Bensona i Faracha. W mojej pracy jest przedstawiony liniowy (czyli optymalny) algorytm dla tekstów nad alfabetem o wielomianowym rozmiarze, opierający się na kilku kombinatorycznych własnościach okresów słów oraz nietrywialnej analizie zamortyzowanej.

Następnie bazując na podobnych (choć rozwiniętych) pomysłach skonstruowałem algorytm o złożoności $\mathcal{O}(n \log \frac{N}{n} + m)$ dla bardziej ogólnego problemu wyszukiwania wzorca w tekście skompresowanym metodą

LZ. Ta metoda jest szczególnie interesująca z praktycznego punktu widzenia: opierają się na niej popularne programy typu `zip` czy `gzip`. Co więcej, w teorii pozwala ona na **wykładnicze** zmniejszenie długości tekstu, przez co skonstruowanie efektywnego algorytmu wyszukiwania wzorca dla tej metody wydaje się być szczególnie trudne. Najlepszy znany wcześniej algorytm autorstwa Faracha i Thorupa działał w (oczekiwanym) czasie $\mathcal{O}(n \log^2 \frac{N}{n} + m)$. Za ten wynik otrzymałem nagrodę dla najlepszej pracy studenckiej na konferencji ESA 2011.

Być może mniej przydatną z czysto praktycznego punktu widzenia, ale fascynującą pod względem teoretycznym, jest wersja problemu, w której kompresujemy zarówno tekst, jak i wzorzec. W pracy przedstawionej na konferencji STACS 2012 pokazałem, że liniowy algorytm wyszukiwania wzorca w tekście skompresowanym metodą LZW może być uogólniony tak, aby wykrywał wystąpienie skompresowanego (metodą LZW) tekstu w skompresowanym (też metodą LZW) tekście w czasie proporcjonalnym do rozmiaru ich skompresowanej reprezentacji. Wcześniej znane rozwiązanie, podane przez Gąsieńca i Ryttera, wymagało czasu $\mathcal{O}((n+m)\log(n+m))$.

Ostatnim wynikiem, który wchodził w skład rozprawy, był szybki algorytm wyszukiwania wielu wzorców w tekście skompresowanym metodą LZW. Oznaczając przez M sumę długości wszystkich wzorców, poprzednio znane rozwiązanie wymagało czasu rzędu M^2, co dla dużej liczby długich wzorców może być nieakceptowalne. Moje rozwiązanie działa w czasie $\mathcal{O}(n \log M)$ lub $\mathcal{O}(n + M^{1+\epsilon})$, co więcej jest koncepcyjnie prostsze niż wcześniej znane rozwiązania dla jednego wzorca o takich samych czasach działania.

Indeksowanie skompresowanego tekstu. Wyszukiwanie wzorca nie jest oczywiście jedynym naturalnym i przydatnym w praktyce sposobem przetwarzania skompresowanych danych. Równie interesujące (i być może nawet lepiej uzasadnione z czysto praktycznego punktu widzenia) jest **indeksowanie** tekstu, czyli budowanie struktury danych, która następnie pozwala na szybkie wyszukanie dowolnego wzorca. W trzech pracach napisanych razem z Travisem Gagie, Juhą Kärkkäinen, Yakovem Nekrichem i Simonem J. Puglisi (zaprezentowanymi na konferencjach CCCG 2013, ISAAC 2012 i LATA 2012) podaliśmy nowe sposoby konstruowania takich indeksów dla tekstów skompresowanych gramatykowo.

Konwersja między różnymi metodami kompresji. Biorąc pod uwagę różnorodność rozważanych sposobów kompresji, ciekawe jest także przyjrzenie się temu, jak szybko można przekształcić tekst skompresowany jedną metodą na inną. W zaprezentowanej na konferencji CPM 2013 pracy napisanej razem z Hideo Bannai, Shunsuke Inenagą i Masayukim Takedą skonstruowaliśmy efektywny algorytm konwersji z reprezentacji gramatykowej do LZW. Kluczowa w rozwiązaniu była dynamiczna struktura danych pozwalająca na szybkie wyznaczanie najdłuższego wspólnego prefiksu dwóch ścieżek w drzewie, które krawędzie są etykietowane literami. Może to być przydatne także w innych problemach dotyczących przetwarzania tekstu skompresowanego metodą LZW.

Przybliżone wyszukiwanie wzorca w skompresowanym tekście. Naturalne jest także zastanowienie się nad przybliżonym wyszukiwaniem wzorca w skompresowanym tekście. W pracy przedstawionej na konferencji SPIRE 2012 zajmowałem się wariantem takiego problemu, w którym dla danych dwóch napisów skompresowanych gramatykowo chcemy wyznaczyć ich odległość edycyjną. Wcześniej znane rozwiązanie pozwalało na osiągnięcie czasu działania $\mathcal{O}(nN \log \frac{N}{n})$, podczas gdy naturalną granicą wydaje się być $\mathcal{O}(nN)$. Mój pomysł pozwala na wyznaczenie takiej odległości w czasie $\mathcal{O}(nN\sqrt{\log \frac{N}{n}})$. Najważniejszym elementem rozwiązania był liniowy algorytm mnożenia pewnej specyficznej klasy macierzy przez wektor. Tego typu macierze pojawiają się także w innym problemach związanych z przybliżonym wyszukiwaniem wzorca, więc podana metoda może znaleźć więcej zastosowań. Bazując na pomysłach użytych wcześniej dla dokładnego wyszukiwania wzorca, razem z Damianem Straszakiem podałem szybki algorytm wyszukiwania wzorca z k błędami lub niezgodnościami w tekście skompresowanym metodą LZW.

Oprócz przetwarznaia skompresowanych tekstów, pracowałem także nad kilkoma problemami związanymi z językami formalnymi.

Wykrywanie preudopowtórzeń. W pracach napisanych z Florinem Maneą, Robertem Mercasem, Dirkiem Nowotką i Catalinem Tisenau (przedstawionych na konferencjach STACS 2013 i CiE 2013) podaliśmy efektywne algorytmy wykrywające w tekstach tak zwane pseudopowtórzenia. Są one uogólnieniem klasycznych powtórzeń (takich jak słowa podwójne), w którym za takie same uznajemy słowo i jego obraz

przez ustalony morfizm lub antymorfizm f. Na przykład, jeśli rozważany alfabet to $\{A, C, G, T\}$, a f jest funkcją, która odwraca podane słowo i zamienia wszystkie A na T, T na A, C na G, a G na C, słowo $ACGTTCCA$ jest pseudokwadratem. Dla danego słowa w i funkcji f interesuje nas sprawdzenie, czy w jest pseudopowtórzeniem, lub wygenerowanie wszystkich podsłów w, które są pseudopowtórzeniami. W innej wersji, dla danego słowa w chcemy sprawdzić, czy istnieje funkcja f, dla której w jest pseudopowtórzeniem. Podaliśmy szereg efektywnych algorytmów dla kilku problemów tego typu.

Konwersja unarnych NFA. W pracy przedstawionej na konferencji CIAA 2011 udowodniłem, że mając dany unarny niedeterministyczy automat skończony na n stanach, można efektywnie skonstruować wyrażenie regularne rozmiaru $\mathcal{O}(\frac{n^2}{\log n})$ lub gramatykę bezkontekstową na $\mathcal{O}(\sqrt{n \log n})$ nieterminalach, które opisują ten sam język. Wcześniej znane ograniczenia były równe (odpowiednio) $\mathcal{O}(n^2)$ i $\mathcal{O}(n^{2/3})$. Otrzymałem za ten wynik nagrodę dla najlepszej pracy na konferencji.

Hiperminimalizacja DFA. W pracach napisanych z Arturem Jeżem i Andreasem Malettim zajmowaliśmy się problemem hiperminimalizacji deterministycznych automatów skończonych, w którym dla danego automatu M szukamy jak najmniejszego automat M', który rozpoznaje język $L(M')$ różniący się od $L(m)$ tylko na skończonej liczbie słów. Pokazaliśmy, że tak zdefiniowany problem można rozwiązać w czasie $\mathcal{O}(n \log^2 n)$. Co więcej, w takim samym czasie można skonstruować najmniejszy automat M', którego rozpoznawany język $L(M')$ różni się od $L(M)$ tylko na słowach długości co najwyżej k. Otrzymaliśmy za ten wynik nagrodę dla najlepszej pracy studenckiej na konferencji MFCS 2009. Na konferencji MFCS 2011 uogólniliśmy te wyniki do przypadku, w którym funkcja przejścia M jest częściowa.

Zliczanie słów danej długości w jezyki bezkontekstowym. W pracy napisanej razem z Dalią Krieger, Naradem Rampersad i Jeffreyem Shallitem i przedstawionej na konferencji DLT 2007 badaliśmy, jak trudne jest wyznaczanie asymptotycznego tempa wzrostu funkcji zliczającej różne słowa o danej długości n w języku zdefiniowanym przez automat (deterministyczny lub nie) lub gramatykę bezkontekstową. Podaliśmy efektywny (liniowy dla automatów, wielomianowy dla gramatyk) algorytm, który pozwala na stwierdzenie, czy ta funkcja jest ograniczona przez wielomian, oraz na wyznaczenie najmniejszego możliwego stopnia takiego wielomianu.

Synchronizacja DFA. W pracach napisanych razem z Andrzejem Kisielewiczem badaliśmy pewne aspekty synchronizacji deterministycznych automatów skończonych, gdzie synchronizacja automatu przez słowo oznacza, że stan po jego przeczytaniu nie zależy od wyboru stanu początkowego.

Publikacje

1. Paweł Gawrychowski, Damian Straszak: Beating $O(nm)$ in approximate LZW-compressed pattern matching, *ISAAC 2013*
2. Paweł Gawrychowski, Gregory Kucherov, Yakov Nekrich and Tatiana Starikovskaya: Minimal discriminating words problem revisited, *SPIRE 2013*
3. Travis Gagie, Paweł Gawrychowski, Yakov Nekrich: Heaviest induced ancestors and longest common substrings, *CCCG 2013*
4. Hideo Bannai, Paweł Gawrychowski, Shunsuke Inenaga, and Masayuki Takeda: Converting SLP to LZ78 in almost linear time, *CPM 2013*
5. Paweł Gawrychowski, Florin Manea, Dirk Nowotka: Discovering hidden repetitions in words, *CiE 2013*
6. Paweł Gawrychowski: Alphabetic minimax trees in linear time, *CSR 2013*
7. Paweł Gawrychowski, Florin Manea, Robert Mercas, Dirk Nowotka, and Catalin Tiseanu: Finding pseudo-repetitions, *STACS 2013*
8. Paweł Gawrychowski: Faster algorithm for computing the edit distance between SLP-compressed strings, *SPIRE 2012*
9. Paweł Gawrychowski: Simple and efficient LZW-compressed multiple pattern matching, *CPM 2012* **Best Student Paper**
10. Travis Gagie, Paweł Gawrychowski, Juha Kärkkäinen, Yakov Nekrich, Simon J. Puglisi: A faster grammar-based self-index, *LATA 2012*
11. Paweł Gawrychowski: Tying up the loose ends in fully LZW-compressed pattern matching, *STACS 2011*

12. Travis Gagie, Paweł Gawrychowski, Simon J. Puglisi: Faster approximate pattern matching in compressed repetitive texts, *ISAAC 2011*
13. Paweł Gawrychowski, Artur Jeż, Andreas Maletti: On minimising automata with errors, *MFCS 2011*
14. Paweł Gawrychowski: Pattern matching in Lempel-Ziv compressed strings: fast, simple, and deterministic, *ESA 2011* **Best Student Paper Award**
15. Paweł Gawrychowski: Chrobak normal form revisited, with applications, *CIAA 2011* **Best Paper Award**
16. Paweł Gawrychowski: Optimal pattern matching in LZW compressed strings, *SODA 2011*, pełna wersja przyjęta do *ACM Transactions on Algorithms*
17. Travis Gagie, Paweł Gawrychowski: Grammar-based compression in a streaming model, *LATA 2010*
18. Paweł Gawrychowski, Łukasz Jeż, Artur Jeż: Validating the Knuth-Morris-Pratt failure function, fast and online, *CSR 2010*
19. Paweł Gawrychowski, Marin Gutan, Andrzej Kisielewicz: On the problem of freenes of multiplicative matrix semigroups, *Theoretical Computer Science* 411(7-9):1115-1120 (2010)
20. Paweł Gawrychowski, Artur Jeż: Hyperminimization made efficient, *MFCS 2009* **Best Student Paper Award**
21. Paweł Gawrychowski, Travis Gagie: Minimax trees in linear time with applications, *IWOCA 2009* **Best Student Paper Award**, pełna wersja w *European Journal of Combinatorics* 34(1):82-90 (2013)
22. Paweł Gawrychowski, Dalia Krieger, Narad Rampersad and Jeffrey Shallit: Finding the growth rate of a regular or context-free language in polynomial time, *DLT 2008*, pełna wersja w *International Journal of Foundations of Computer Science* 21(4):597-618 (2010)
23. Jaroslaw Byrka, Paweł Gawrychowski, Katharina T. Huber, Steven Kelk: Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks, *Journal of Discrete Algorithms* 8(1):65-75 (2010)
24. Paweł Gawrychowski, Andrzej Kisielewicz: 2-Synchronizing Words, *LATA 2008*
25. Alessandra Cherubini, Paweł Gawrychowski, Andrzej Kisielewicz and Brunetto Piochi: A Combinatorial Approach to Collapsing Words, *MFCS 2006*