

dr inż. Zbigniew Kokosiński
Politechnika Krakowska
Wydział Inżynierii Elektrycznej i Komputerowej
Katedra Automatyki i Technik Informatycznych
ul. Warszawska 24, 31-155 Kraków
tel. 12-628-2685
email: zk@pk.edu.pl

ZAŁĄCZNIK 2 PL

ResearcherID: K-4996-2013
ORCID ID <https://orcid.org/0000-0002-2082-9401>

Autoreferat

przedstawiający opis dorobku i osiągnięć naukowych

Kraków - maj 2018

Wykształcenie i przebieg zatrudnienia

1. Nazwisko i imię : Zbigniew Kokosiński
Data i miejsce urodzenia: 23 czerwca 1957, Kraków
2. Posiadane dyplomy i stopnie naukowe:
 - a) 1992 – doktor nauk technicznych (dyplom z wyróżnieniem), dyscyplina : informatyka, specjalność: systemy przetwarzania równoległego, Wydział Elektroniki¹ Politechniki Gdańskiej, promotor: prof. dr inż. Tadeusz Puchałka (Politechnika Poznańska), tytuł rozprawy: „Układy generatorów obiektów kombinatorycznych dla systemów sekwencyjnych i równoległych”.
 - b) 1982 – magister inżynier transportu, specjalność : sterowanie ruchem w transporcie, Wydział Transportu² Politechniki Krakowskiej, promotor: dr inż. Adam Kaprański (PK), tytuł pracy: „Porównanie efektywności algorytmów syntezy kombinacyjnych układów logicznych. Postać dysjunkcyjna”.
3. Informacje o dotychczasowym zatrudnieniu w jednostkach naukowych:
 - a) 1992 – obecnie, adiunkt naukowo-dydaktyczny
Katedra Automatyki i Technik Informacyjnych³, Wydział Inżynierii Elektrycznej i Komputerowej⁴ Politechniki Krakowskiej
 - b) 2006-2007 – adiunkt
Wyższa Szkoła Europejska im. ks. Józefa Tischnera w Krakowie (1/8 etatu).
 - c) 1994-1997 – Assistant Professor
Mathematical Foundations of Computer Science Laboratory, Department of Computer Software, The University of Aizu, Aizu-Wakamatsu, Japonia.
 - d) 1986-1992 – starszy asystent naukowo-dydaktyczny,
1983-1986 – asystent naukowo-dydaktyczny
1982-1983 – asystent stażysta
Instytut Elektrotechniki i Elektroniki, Wydział Transportu⁵ Politechniki Krakowskiej

¹ 1996- Wydział Elektroniki, Telekomunikacji i Informatyki

² 1997- Wydział Inżynierii Elektrycznej i Komputerowej

³ 1991-1997 Instytut Automatyki

⁴ 1991-1997 Wydział Inżynierii Elektrycznej

⁵ 1988-1991 Wydział Inżynierii Transportowej i Elektrycznej

Osiągnięcie naukowe w postaci cyklu publikacji

4. Wskazanie osiągnięcia po otrzymaniu stopnia doktora, wynikające z art. 16 ust. 2 ustawy z dnia 14 marca 2003 r. o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki (Dz. Ustaw nr 65, poz. 595 ze zm.):

A. Osiągnięciem stanowiącym przedmiot tego wniosku jest cykl powiązanych tematycznie publikacji zatytułowany:

Efektywne równoległe algorytmy generacji obiektów kombinatorycznych w modelach: asocjacyjnym, sieci połączeniowej i populacyjnym

1. [B8] Kokosiński Z.: *On parallel generation of t -ary trees in an associative model*, Postproc. PPAM'2001 (Parallel Processing and Applied Mathematics), Lect. Notes Comput. Sci. 2328, 228-235 (2002)
(wkład własny 100%, lista **JCR**, IF=**0,515**)
2. [B11] Kokosiński Z.: *On generation of permutations through suffix/prefix reversing in a cellular network*, Postproc. PPAM'2003 (Parallel Processing and Applied Mathematics), Lect. Notes Comput. Sci. 3019, 249-254 (2004)
(wkład własny 100%, lista **JCR**, IF=**0,513**)
3. [B13] Kokosiński Z.: *A parallel dynamic programming algorithm for unranking t -ary trees*, Postproc. PPAM'2003 (Parallel Processing and Applied Mathematics), Lect. Notes Comput. Sci. 3019, 255-260 (2004)
(wkład własny 100%, lista **JCR**, IF=**0,513**)
4. [B15] Kokosiński Z., Kołodziej M., Kwarciany K.: *Parallel genetic algorithm for graph coloring problem*, Proc. ICCS'2004 (Computational Science), Lect. Notes in Comput. Sci. 3036, 217-224 (2004)
(wkład własny 60%, lista **JCR**, IF=**0,513**)
5. [B17] Kokosiński Z., Kwarciany K., Kołodziej M.: *Efficient graph coloring with parallel genetic algorithms*, Computing and Informatics 24, 123-147 (2005)
(wkład własny 70%, lista **JCR**, IF=**0,091**)
6. [B18] Kokosiński, Z.: *Effects of versatile crossover and mutation operators on evolutionary search in partitions and permutation problems*, Intelligent Information Systems: New Trends in Intelligent Information Processing and Web Mining 2005, Proc. of the Int. Conference, IIS:IIPWM'2005, Advances in Soft Computing, Springer, 299–308 (2005)
(wkład własny 100%, lista **Web of Science**)
7. [B19] Kokosiński Z.: *A new algorithm for generation of exactly m -block partitions in associative model*, Postproc. PPAM'2005 (Parallel Processing and Applied Mathematics), Lect. Notes Comput. Sci. 3911, 67-74 (2006)
(wkład własny 100%, lista **JCR**, IF=**0,402**)

8. [B23] Kokosiński Z.: *On generation of derangements, partial derangements and permutations*, Postproc. PPAM'2007 (Parallel Processing and Applied Mathematics), Lect. Notes Comput. Sci. 4967, 219-228 (2008)

(wkład własny 100%, lista **Web of Science**)

9. [B26] Kokosiński Z.: *Parallel enumeration of t-ary trees in ASC SIMD model*, Int. Journal of Computer Science and Network Security, Vol. 11, No. 12, 38–49 (2011)

(wkład własny 100%, lista **B MNiSW**)

10. [B30] Kokosiński Z.: *A parallel dynamic programming algorithms for unranking set partitions*, Technical Transactions, 4-AC/2013, issue 28, year 110, 29–38 (2013)

(wkład własny 100%, lista **B MNiSW**)

11. [B39] Kokosiński Z.: *On generation of permutations of m out of n elements*, Information Processing Letters 124, 1-5 (2017)

(wkład własny 100%, lista **JCR**, IF=**0,748**)

Cykl składa się z 11 wybranych publikacji (w tym 7 na liście JCR), stanowiących samodzielny dorobek kandydata po otrzymaniu stopnia doktora, spośród 22 publikacji związanych z tym tematem badawczym. Sumaryczny IF cyklu wynosi 3,295, a ważony IF=3,063. Według bazy Web of Science Core Collection publikacje wchodzące w skład cyklu były cytowane 29 razy w 21 artykułach (25 razy w 17 niezależnych artykułach).

Uwaga: numeracja pozycji cyklu jest podwójna : koresponduje z numeracją cyklu w Załączniku 3 (wykaz publikacji) oraz z numeracją publikacji habilitanta w porządku chronologicznym, przyjętym w Autoreferacie (wykaz na s. 32-36).

B. Cel naukowy podjętych badań oraz miejsce wskazanego cyklu publikacji w kontekście całości badań habilitanta związanych z tematyką cyklu

Badania nad obiektami kombinatorycznymi, ich generacją sekwencyjną, równoległą i populacyjną, enumeracją oraz funkcjami numeracyjnymi (rankingowymi) odgrywają duże znaczenie w rozwoju narzędzi informatycznych stosowanych w kombinatoryce [5,8,9,20], optymalizacji dyskretnej [3,10,17], algorytmach randomizowanych [15] oraz w przetwarzaniu równoległym problemów należących do różnych klas złożoności [4,6]. Nieprzerwane badania w tym obszarze mają charakter podstawowy dla informatyki. Do najważniejszych prac na temat generacji konfiguracji kombinatorycznych należą m.in. książki D.E. Knutha [8], F. Ruskey'a [16], S.G. Williamsona [20], a na polskim gruncie książka W. Lipskiego [11]. Generacja populacyjna w kontekście algorytmów genetycznych została omówiona w zarysie w książce Z. Michalewicz [13].

Cykl publikacji wybrany w ramach *osiągnięcia naukowego* obejmuje wybrane prace poświęcone równoległej generacji obiektów kombinatorycznych w dwóch modelach obliczeń, algorytmom numeracyjnym oraz generacji populacyjnej obiektów kombinatorycznych w algorytmach ewolucyjnych stosowanych w optymalizacji dyskretnej.

Cel naukowy badań podjętych przez habilitanta miał charakter zarówno poznawczy jak i praktyczny. Badania miały na celu lepsze zrozumienie właściwości klas obiektów kombinatorycznych i relacji pomiędzy tymi klasami, konstrukcję algorytmów generacyjnych

i rankingowych o niskiej złożoności obliczeniowej, uwzględnienie wpływu reprezentacji obiektów oraz porządku określonego na zbiorze obiektów na konstrukcję i złożoność algorytmów (por. [18]). W celu ujednoczenia opisu, do modelowania obiektów kombinatorycznych wykorzystano konsekwentnie pojęcie funkcji wyboru (systemu reprezentantów) rodziny indeksowanej zbiorów [12,14], spełniającej warunki określone dla danego typu obiektu. Odkryte właściwości obiektów stały się punktem wyjścia do konstrukcji algorytmów sekwencyjnych i równoległych. Konstrukcję algorytmów numeracyjnych (konwersja: *obiekt* \rightarrow *liczba porządkowa* i *liczba porządkowa* \rightarrow *obiekt*) opartych na technice programowania dynamicznego poprzedziło wyznaczenie równań rekurencyjnych, określających zależności pomiędzy stabularyzowanymi współczynnikami reprezentującymi odległości pomiędzy pewnymi funkcjami wyboru w określonym porządku liniowym. Założonym walorem aplikacyjnym miała być możliwość implementacji sprzętowej równoległych algorytmów generacyjnych (z wyjątkiem generacji populacyjnej) aby umożliwić budowę szybkich generatorów sprzętowych i ich wykorzystanie w przetwarzaniu równoległym.

Pierwsze badania habilitanta związane z tematyką cyklu, prowadzone w latach 1990-1994, rozpoczęte od publikacji [A7] poświęconej generacji permutacji, zaowocowały rozprawą doktorską [A8], w której na bazie algorytmów generacji permutacji, kombinacji i podziałów, zaprojektowanych w modelu sieci połączeniowej oraz algorytmów rankingowych dla tych obiektów, opracowany został programowalny generator sprzętowy trzech typów obiektów kombinatorycznych umożliwiający wybór typu obiektu, zaprogramowanie pierwszego obiektu sekwencji oraz jej długości. Układ generatora permutacji został następnie opatentowany [AP2].

W kolejnych latach po doktoracie skonstruowane zostały algorytmy generacji różnych obiektów kombinatorycznych (kombinacji, kompozycji liczb, podziałów, drzew, permutacji, nieporządków, wariacji), a także pewnych ich odmian, w różnych modelach obliczeniowych (sieci połączeniowej, modelu asocjacyjnym, tablicy systolicznej), w różnych reprezentacjach i porządkach liniowych. Skonstruowano także liczne algorytmy konwersji *obiekt* \rightarrow *liczba porządkowa* i *liczba porządkowa* \rightarrow *obiekt*. Uzyskane wyniki mogą być zastosowane w przetwarzaniu równoległym i rozproszonym [7], kodowaniu enumeracyjnym [19], kryptografii itp.

Rozszerzenie zainteresowań badawczych habilitanta o metaheurystyki i problematykę optymalizacji kombinatorycznej zaowocowało sformułowaniem kolejnych celów, takich jak badanie równoległych i hybrydowych metaheurystyk dla trudnych problemów obliczeniowych [2] oraz nowych mechanizmów generacji populacyjnej obiektów kombinatorycznych w zastosowaniu do wybranych problemów permutacyjnych i podziałowych, np. kolorowania grafów (klasyczne, sumacyjne, odporne) [10,21], szeregowania zadań (OOSP), probabilistycznego problemu komiwojażera (PTSP) itp.

Algorytmy ewolucyjne wykorzystano jako użyteczne narzędzie do empirycznego poszukiwania oszacowań takich charakterystyk grafów jak suma chromatyczna $\sum(G)$ [10], liczba sumy chromatycznej $s(G)$ [10] oraz liczba podziałowa $\psi_k(G)$, $k \geq 2$ [B40], przyczyniając się w wielu przypadkach do dokładniejszej i pełniejszej charakterystyki grafów z repozytorium DIMACS [22].

Tablica I zawiera zestawienie algorytmów generacji obiektów kombinatorycznych opracowanych przez habilitanta w latach 1990-1994 oraz 1995-2017.

Tablica II zawiera zestawienie opracowanych algorytmów rankingowych.

Tablica III zawiera zestawienie algorytmów ewolucyjnych, w których zastosowano nowe mechanizmy generacji populacyjnej wybranych obiektów kombinatorycznych inspirowane zastosowaniami (krzyżowanie, hybrydyzacja), opracowanych i przetestowanych w latach 1999-2016.

W tablicy IV przedstawiono wybrane aplikacje uzyskanych wyników. Pogrubieniem w Tablicach I-IV zaznaczono pozycje literaturowe należące do wybranego cyklu publikacji.

Prace wskazane w wybranym cyklu publikacji są reprezentatywne dla całości dorobku habilitanta w tym obszarze, zarówno jeśli chodzi o różnorodność tematyczną, miejsca publikacji, jak i liczbę cytowań.

Tablica I Algorytmy generacji obiektów kombinatorycznych w dorobku habilitanta

Obiekt kombinatoryczny	Algorytm generacyjny	Złoż. oblicz./ obiekt	Porządek generacji sekwencji sterującej	Porządek generacji sekwencji wynikowej	Model obliczeniowy	Źródło literatur.
permutacja	PG (Permutation Generation)	O(n)	leksykograficzny (dowolny zakres)	liniowy (2 sekwencje)	sieć łącz.	[A7, AP2]
permutacja k -podzbiór podział	Algorytm 5, GPCP	O(n)	leksykograficzny leksykograficzny leksykograficzny	liniowy leksykograficzny leksykograficzny	sieć łącz.	[A8, A10]
permutacja	Algorytm 6	O(n)	odwrotny leksyk.	liniowy (2 sekwencje)	sieć łącz.	[A8]
kompozycja liczby całkowitej	COMPGEN1 COMPGEN-PAR1 COMPGEN2 COMPGEN-PAR2	O(1)		leksykograficzny malejący	tablica liniowa	[B3]
k -podzbiór	COMBGEN BINCOMBGEN	O(1)	leksykograficzny leksykograficzny	leksykograficzny odwrotny leksykograf. (reprezent. binarna)	ASC SIMD	[B4, B16]
podział	PARTGEN	O(1)		leksykograficzny	ASC SIMD	[B6]
drzewa t -narne	Z-TREEGEN (TREEGEN)	O(1)	leksykograficzny (t -sekwencje)	leksykograficzny (z -sekwencje)	ASC SIMD	[B8, B26]
drzewa t -narne	X-TREEGEN	O(1)	leksykograficzny (t -sekwencje)	leksykograficzny malejący (x -sekwencje)	ASC SIMD	[B26]
permutacja	PERMGEN	O(n)	leksykograficzny	liniowy (suffix inversion) liniowy (prefix inversion)	sieć łącz.	[B11]
podział m -blokowy	M-PARTGEN	O(1)		leksykograficzny	ASC SIMD + licznik równoległy	[B19]
nieporządek częściowy nieporządek	PDGENLEX PDGENREVLEX	O(kn)	leksykograficzny odwrotny leksyk.	liniowy (2 sekwencje) liniowy (2 sekwencje)	sieć łącz.	[B23]
(n,m) -permut. (wariacja)	VARGEN VARGEN-NR	O(m)	leksykograficzny leksykograficzny (dowolny zakres)	liniowy (2 sekwencje) liniowy (2 sekwencje)	sieć łącz.	[B39]

Tablica II Algorytmy numeracyjne dla obiektów kombinatorycznych w dorobku habilitanta

Obiekt kombinatoryczny	Algorytm konwersji liczba porządkowa \rightarrow obiekt	Złożoność oblicz./obiekt	Metoda/własność	Źródło literatur.
permutacja	UNRANKING	O(n^2)		[A7]
permutacja k -podzbiór podział	Algorytm 1, Algorytm 2 Algorytm 3 Algorytm 4	O(n) O(n) O(n)	(porz. leks. i odwrotny leks.) program. dynamiczne program. dynamiczne	[A8]
k -podzbiór	UNRANKCOMB-A UNRANKCOMB-B UNRANKCOMB-C UNRANKCOMB-D	O(n) O(n) O(klogn) O(n)	program. dynamiczne program. dynamiczne program. dyn. + wyszukiw. binarne „dziel i zwyciężaj” j.w.	[B1]
k -podzbiór z powtórzeniami	UNRANKCOMB-D modyfikacje w/w	O(n)		
k -podzbiór	UNRANKCOMB-E	O(k)	program. dyn. + przetw. asocjacyjne	[B2]
kompozycja liczby całkowitej	UNRANKCOMB + konwersja k -podzbiór \rightarrow kompozycja	O(n)	w zależności od wersji algorytmu UNRANKCOMB	[B3]
drzewo t -narne	UNRANKTREE UNRANKTREE-PAR	O(nt) O(n)	program. dynamiczne program. dyn. + przetw. asocjacyjne	[B13, B26]
podział	UNRANKPART UNRANKPART-PAR	O(n^2) O(n)	program. dynamiczne program. dyn. + przetw. asocjacyjne	[B30]
(n,m) -permut. (wariacja)	UNRANKVAR	O(m)	program. dynamiczne	[B39]

Tablica III Algorytmy generacji populacyjnej w dorobku habilitanta

Obiekt kombinatoryczny	Algorytm generacyjny	Reprezentacja obiektu	Operator krzyżowania	Optymalizacja lokalna	Złoż. oblicz./obiekt	Źródło literaturowe
permutacja	ewolucyjny	wektorowa (Coset)	Klasyczny		O(n)	[B7, B18]
permutacja z powtórzeniami	ewolucyjny (hybrydowy)	wektorowa	LOX	LPT-Task LPT-Machine	O(nm)	[B21]
podział	ewolucyjny	blokowa	SPPX		O(1)	[B15, B17, B18]
podział	ewolucyjny	wektorowa	CEX		O(n)	[B15, B17]

Tablica IV Wybrane zastosowania algorytmów generacyjnych i numerycznych

Obszar zastosowania	Rodzaj zastosowania	Źródło literaturowe
Systemy równoległe i rozproszone	Podział zadania generacji w wyszukiwaniu i testowaniu wyczerpującym	[A8, A9, B1, B2, B13, B26, B30, B39]
	Szybka generacja obiektów w modelach niskiego poziomu, umożliwiających implementację w technologii VLSI/ FPGA	[A7, A8, B3, B4, B6, B8, B11, B16, B19, B23, B24, B39]
	Sprzętowa generacja masek i komparandów (procesory asocjacyjne, DSM [7])	[A10, B4, B9, B12, B16, B25]
Generacja indeksów elementów zbioru danych	Kodowanie enumeracyjne [19]	[A7, A8, B1, B2, B13, B26, B30, B39]
Kryptografia	Szyfrator oparty na sprzętowym generatrice nieporządków	[B31]
Metaheurystyki	Generacja populacji pseudolosowej (inicjalizacja albo reinicjalizacja) [13,17]	[A7, A8, B1, B2, B13, B26, B30, B39]
	Operacje krzyżowania osobników w algorytmach ewolucyjnych	[B15, B17, B18, B20, B32, B37]
Teoria grafów	Empiryczne oszacowania górne charakterystyk grafów DIMACS : sumy chromatycznej $\sum(G)$, liczby sumy chromatycznej $s(G)$, liczby podziałowej $\psi_k(G)$, $k \geq 2$.	[B20, B29, B37, B40]

Literatura:

- [1] Akl S.G., Stojmenović I.: *Generating combinatorial objects on a linear array of processors*, [in] Zomaya A.Y. (ed): *Parallel Computing: Paradigms and Applications*, 639–670, International Thomson Computer Press, 1996.
- [2] Alba, E. (Ed.): *Parallel metaheuristic – a new class of algorithms*, John Wiley & Sons, 2005
- [3] Ausiello G. et al.: *Complexity and approximation. Combinatorial optimization problems and their approximability properties*, Springer 1999
- [4] Garey, R., Johnson, D. S.: *Computers and intractability. A guide to the theory of NP-completeness*, Freeman, 1979.
- [5] Graham, R.L., Knuth D.E., Patashnik O.: *Matematyka konkretna*, PWN, Warszawa 1996.
- [6] Greenlaw, R., Hoover H.J.: *Limits to parallel computations : P-completeness theory*, Oxford University Press, 1995
- [7] Kapralski, A.: *Sequential and parallel processing in depth search machines*, World Scientific, 1994
- [8] Knuth, D.E.: *The art of computer programming*, Vol. 4A *Combinatorial algorithms*, Part 1, Addison-Wesley 2011.
- [9] Kreher D.L., Stinson D.R.: *Combinatorial Algorithms, Generation, Enumeration and Search (CAGES)*. Chapter 2: *Generating Elementary Combinatorial Objects*, CRC Press, 1999, 31–32.
- [10] Kubale, M.: (Red.): *Optymalizacja dyskretna. Modele i metody kolorowania grafów*, WNT, Warszawa, 2002.
- [11] Lipski W.: *Kombinatoryka dla programistów*, WNT, Warszawa 1982.
- [12] Lipski W., Marek W.: *Analiza kombinatoryczna*, PWN, Warszawa 1986.

- [13] Michalewicz Z.: *Evolutionary Algorithms + Data Structures = Evolutionary Programs*, 3rd ed., Springer, 1996.
- [14] Mirsky L. : *Transversal theory*, Academic Press, New York 1971.
- [15] Motwani R., Raghavan P.: *Randomized algorithms*, Cambridge University Press 1995.
- [16] Ruskey, F.: *Combinatorial generation*, Working Version (1j-CSC 425/520), <http://www.edu/~ruskey/book.pdf> , 2003.
- [17] Sait S.M., Youssef H.: *Iterative computer algorithms with applications in engineering*, IEEE Computer Society, Los Alamitos 1999.
- [18] Sedgewick R.: *Permutation generation methods*, ACM Computing Surveys 9, Issue 2, 1977, 137–164.
- [19] Tomič R.V.: *Quantized indexing: background information*, 1stWorks Corporation Technical Report TR05-0625, 1st Works Corporation, June 25, 2005, 39 pp. (www.1stWorks.com).
- [20] Williamson S.G.: *Combinatorics for computer science*, Computer Science Press, Rockville 1985.
- [21] Yáñez J., Ramirez J.: *The robust coloring problem*, European Journal of Operational Research, Vol. 148, No. 3, 2003, 546–558.
- [22] DIMACS ftp site. Available at: <ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/> .

C. Osiągnięte wyniki badań i ich aplikacje

C.1. Wprowadzenie

Generacja sekwencji obiektów kombinatorycznych reprezentowanych przez słowa kodowe (*codewords*) jest analogiczna do procesu zliczania w pewnym kodzie. Najprostszym modelem obliczeniowym dla tego zadania jest (złożony) licznik równoległy. Użyteczność tego modelu jest ograniczona do niektórych klas obiektów i niektórych reprezentacji. R. Sedgewick wykazał, że wszystkie metody generacji permutacji zawierają *implicite* tę samą sekwencję sterującą (*factorial counting*) [3]. Spostrzeżenie to wykorzystano w części algorytmów generacyjnych do rozróżnienia pomiędzy sekwencją sterującą a sekwencją wynikową.

W prowadzonych badaniach zdecydowano się na jednolitą reprezentację obiektów w postaci funkcji wyboru (systemu reprezentantów, transwersali) rodziny indeksowanej zbiorów, umożliwiającą dowolne modelowanie różnych typów obiektów.

W algorytmach generacyjnych i rankingowych istotną rolę odgrywa porządek generacji: leksykograficzny, antyleksykograficzny, odwrotny leksykograficzny, ko-leksykograficzny, Gray'a, inne porządki liniowe oraz porządek pseudolosowy. Liniowy porządek generacji sekwencji sterującej/wynikowej umożliwia konstrukcję algorytmów rankingowych. Jedną z metod stosowanych w konstrukcji algorytmów konwersji *liczba porządkowa* \rightarrow *obiekt kombinatoryczny* jest programowanie dynamiczne. Algorytmy te pozwalają na równomierny podział zadania generacji danego zbioru obiektów na podzbiory odpowiadające przedziałom indeksów (*load balancing*) i ich równoległą generację w systemach wieloprocesorowych. Generator liczb losowych i algorytm konwersji *liczba porządkowa* \rightarrow *obiekt kombinatoryczny* stwarzają z kolei możliwość generacji losowej danej klasy obiektów.

Akl i I. Stojmenović [1] sformułowali w swoich pracach szereg postulatów pod adresem równoległych algorytmów generacji obiektów kombinatorycznych : 1) leksykograficzny porządek generowanych obiektów – w wielu przypadkach dowolny porządek liniowy jest również akceptowalny; 2) koszt-optymalność algorytmu równoległego: iloczyn liczby procesorów n użytych w algorytmie i czasu obliczeń odpowiada - z dokładnością do stałego współczynnika – dolnej granicy liczby operacji niezbędnej do rozwiązania problemu, zwykle uwzględnia się również czas generacji wyjścia; 3) stały czas pomiędzy dwoma kolejnymi obiektami – własność ważna tam, gdzie generowane obiekty są przedmiotem dalszego przetwarzania, tzn. są wejściem dla dalszych obliczeń; 4) model obliczeń równoległych jest tak prosty, jak to jest możliwe – równoległe procesory są indeksowane od 1 do n , posiadają proste wzory połączeń/komunikacji, a model obliczeń jest odpowiedni do implementacji sprzętowej w technologii VLSI/FPGA; 5) procesory nie wymagają

dużej pamięci, najchętniej stałą liczbę rejestrów, każdy o długości $\Theta(\log n)$ bitów, dla przechowywania stałych i zmiennych algorytmu generacji; 6) algorytm powinien generować pełny zbiór obiektów – jest to ważne we wszystkich aplikacjach opartych o wyczerpujące wyszukiwanie lub testowanie – w niektórych przypadkach dopuszcza się losową generację niepełnego zbioru obiektów.

Ponieważ generowane obiekty kombinatoryczne danego typu podlegają często dalszemu przetwarzaniu, sformułowano dodatkowe postulaty: 7) algorytm generacji jest dostatecznie szybki w stosunku do algorytmu przetwarzania; 8) reprezentacja generowanych obiektów odpowiada reprezentacji stosowanej na etapie przetwarzania, aby uniknąć niepotrzebnej konwersji pomiędzy reprezentacjami [A8].

W algorytmach generacji obiektów kombinatorycznych objętych cyklem publikacji wykorzystano równoległe modele obliczeń: model asocjacyjny (ASC SIMD) i sieć połączeniową (*interconnection network*), a poza cyklem publikacji – tablicę systoliczną (*linear array*). Wszystkie te modele zapewniają możliwość implementacji generatorów w układach VLSI/FPGA, wychodząc na przeciw potrzebie sprzętowej akceleracji obliczeń w krytycznych zastosowaniach.

W iteracyjnych algorytmach populacyjnych (ewolucyjnych, mrówkowych, pszczelich, rojowych itp.) zbiór obiektów kombinatorycznych reprezentuje często zbiór rozwiązań problemów optymalizacyjnych. W kolejnych iteracjach mamy do czynienia z generowaniem populacyjnym obiektów kombinatorycznych, według pewnych ustalonych reguł, potencjalnych członków następnej populacji (podpopulacji), reprezentujących rozwiązania dopuszczalne, które są oceniane i dalej przetwarzane. Najczęściej projektowanymi mechanizmami generacji populacji są dedykowane operatory rekombinacji.

Do podstawowych zastosowań metaheurystyk populacyjnych należy przybliżone rozwiązywanie dyskretnych i ciągłych problemów optymalizacyjnych. Obserwowany jest stały rozwój tych metod związany m.in. z nowymi zastosowaniami oraz technikami zrównoleglania i hybrydyzacji. Osiągnięcia matematyki dyskretniej i informatyki stymulują się wzajemnie. Jednym z impulsów rozwojowych na styku tych dziedzin są specjalne tematy badawcze ogłaszane przez The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) – por. <http://dimacs.rutgers.edu/SpecialYears> . Do trudnych problemów obliczeniowych, dla których poszukiwano tą drogą efektywnych technik obliczeniowych zaliczono m. in. problemy NP-zupełne: wyznaczenia maksymalnej kliki w grafie (CLIQUE), kolorowania grafu (GCP) oraz spełnialności formuł boolowskich (SAT) [2]. Dla celów porównawczych zgromadzono obszerny zbiór benchmarków. Charakterystyka grafów testowych DIMACS dla GCP za pomocą liczb reprezentujących ich kombinatoryczne właściwości jest jednym z zastosowań prac habilitanta wchodzących w skład cyklu publikacji.

Literatura

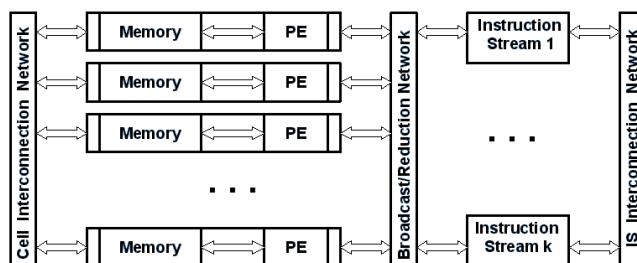
- [1] Akl S.G., Stojmenović I.: *Generating combinatorial objects on a linear array of processors*, [in] Zomaya A.Y. (ed): *Parallel Computing: Paradigms and Applications*, 639–670, International Thomson Computer Press, 1996.
- [2] Johnson D.S., Trick M.A. (eds.): *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26, American Mathematical Society (1996)
- [3] Sedgewick R.: *Permutation generation methods*, ACM Computing Surveys 9, Issue 2, 1977, 137-164.

C.2. Algorytmy generacyjne w modelu asocjacyjnym [B8, B19, B26]

Równoległe modele obliczeniowe różnią się między sobą znacznie pod względem uniwersalności i stopnia złożoności. Z praktycznego punktu widzenia uzasadnione jest stosowanie dedykowanych modeli zapewniających efektywną implementację kluczowych operacji algorytmu

rozwiązującego dany problem. W generacji kombinatorycznej wystarczające są proste modele obliczeń lub ich kombinacje zapewniające generację kolejnych obiektów, najlepiej w stałym czasie $O(1)$.

W równoległych algorytmach generacyjnych, rozpatrywanych poniżej, dominują operacje typu *broadcast* (*multicast*) oraz *maximum*. Pierwsza z nich służy do wysyłania danych z jednego źródła do zbioru (podzbioru) wybranych destynacji, a druga do wyznaczania maksimum zbioru n -elementowego. W popularnym modelu CREW PRAM koszt powyższych operacji jest rzędu $O(\log n)$. W równoległych modelach wyposażonych w magistrale, takich jak PRAM [4], BSR (*Broadcasting with Selective Reduction*) [4], LARPBS [17], koszt ten można obniżyć do $O(1)$. Jednak bardziej odpowiednim modelem jest skalowalny procesor asocjacyjny [9,29]. Spośród wielu modeli o zbliżonych własnościach, klasyfikowanych jako podklasa modelu SIMD, wybrany został dobrze udokumentowany model ASC (*ASsociative Computing*) [5,18,19,29]. W modelu ASC instrukcje wykorzystują równoległość danych (*data parallelism*), a asocjacyjne wyszukiwanie wzorca i operacje wyboru *minimum* i *maximum* są wykonywane w stałym czasie. Synchronizację strumienia instrukcji zapewnia równoległe sterowanie. W stosunku do modelu tablicy liniowej procesorów zaprogramowanie asocjacyjnych procesorów równoległych generujących w sposób równoległy zbiór obiektów jest znacznie prostsze ze względu na mniejszą liczbę wykorzystywanych zmiennych (rejestrów). Model procesora ASC SIMD jest zilustrowany na rys. 1, a jego dokładniejszy opis zawiera praca [19]. W większości prac zawierających generacyjne algorytmy asocjacyjne do opisu algorytmów wystarcza uproszczony model ASC.



Rys. 1. Model procesora asocjacyjnego ASC SIMD [19].

W literaturze przedmiotu poczesne miejsce zajmuje generacja struktur drzewiastych [1-3,7,8,13,20-22,24-28,30,31].

Do cyklu publikacji włączone zostały artykuły [B8,B26] zawierające algorytmy generacyjne dla drzew t -narnych w modelu ASC SIMD. Umożliwiają one zrównoleglenie obliczeń na poziomie pojedynczego obiektu, spełniając wszystkie postulaty S.G. Akł'a i I. Stojmenovič'a [4]. Ponadto włączony został artykuł o generacji podziałów m -blokowych w modelu mieszanym [B19].

W pracy [B8] przedstawiono konstrukcję algorytmu asocjacyjnego TREEGEN, generującego drzewa t -narne w postaci tzw. z -sekwencji [31]. Sekwencję sterującą tworzą wprowadzone w tej pracy po raz pierwszy tzw. t -sekwencje, których generacja w modelu asocjacyjnym jest prostsza niż sekwencji wyjściowej. Generacja pojedynczego obiektu odbywa się w stałym czasie.

W pracy [B26] przedstawiono asocjacyjny algorytm BINTREEGEN, generujący drzewa t -narne w postaci tzw. x -sekwencji (wektorów binarnych) [32]. Kolejne drzewa są generowane w czasie $O(1)$, w malejącym porządku leksykograficznym. Występuje w tym przypadku analogia do algorytmu BINCOMBGEN [B4], generującego w postaci binarnej kombinacje. Generatory obiektów w postaci binarnej mogą zostać zastosowane do sprzętowej generacji wzorców i/lub masek w procesorach asocjacyjnych [10,11,18]. Nie wymagają kosztownej konwersji pomiędzy reprezentacjami.

W pracy [B19] opublikowany został częściowo asocjacyjny algorytm do generowania dokładnie m -blokowych podziałów zbioru n -elementowego. W konstrukcji algorytmu skojarzono

dwa modele: asocjacyjny i licznika równoległego. Asocjacyjny algorytm COMBGEN [B4] stanowi część algorytmu generacyjnego M-PARTGEN i odpowiada za generację częściowych funkcji wyboru κ (k -podzbiorów) – generacja (n,k) -kombinacji reprezentowanych przez rosnące funkcje wyboru odbywa się w porządku leksykograficznym – a algorytm COMBGEN generuje kolejne obiekty w stałym czasie. Z kolei w modelu licznika równoległego generowane są częściowe funkcje wyboru α . Konkatenacja częściowych funkcji wyboru κ i α daje wynikową funkcję wyboru ρ . Poza tym hybrydowym modelem nie jest znany algorytm w pełni asocjacyjny dla tej klasy obiektów. Poszczególne obiekty są generowane w stałym czasie, a specyficzna budowa algorytmu sprzyja zrównolegleniu obliczeń na poziomie zbioru obiektów. Problemem do rozwiązania jest równomierne obciążenie generacją zbioru procesorów o dowolnej – w pewnym zakresie – liczebności.

Równoległe algorytmy generacyjne w modelu asocjacyjnym są w pełni oryginalnym osiągnięciem habilitanta.

Literatura:

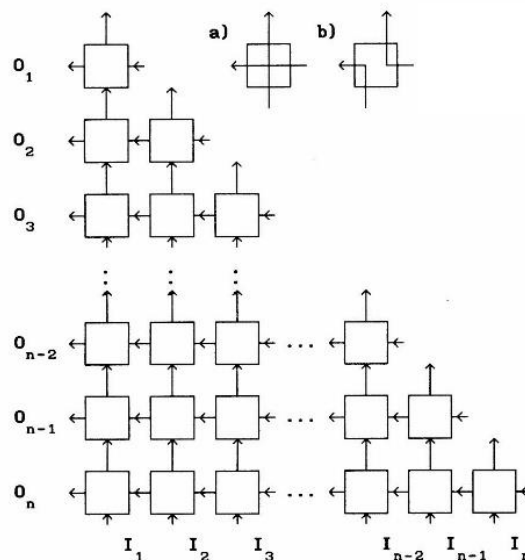
- [1] Ahrabian, H., Nowzari-Dalini A., Salehi, E.: *Gray code algorithm for listing k -ary trees*, Studies in Informatics and Control, 2004, 243–251.
- [2] Ahrabian, H., Nowzari-Dalini A.: *Parallel generation of t -ary trees in A -order*, The Computer Journal, 2007, 581–588.
- [3] Akl, S.G., Stojmenović, I.: *Generating t -ary trees in parallel*, Nordic J. of Computing, 1996, 63–71.
- [4] Akl, S.G.: *Parallel computation: models and methods*, Prentice Hall, 1997, 475–509.
- [5] *ASC Research Papers*, <http://www.cs.kent.edu/~potter/research/papers/>
- [6] Butler J.T., Sasao T.: *Index to Constant Weight Codeword Converter*, Proc. Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Comp. Sci. 6578, 193–205, 2011.
- [7] Er, M.C.: *Lexicographic listing and ranking t -ary trees*, The Computer Journal, 1987, 559–572.
- [8] Er, M.C.: *Efficient generation of k -ary trees in natural order*, The Computer Journal, 1992, 306–308.
- [9] Foster, C.C.: *Content addressable parallel processors*, Van Nostrand Reinhold, 1976.
- [10] Kapralski, A.: *Sequential and parallel processing in depth search machines*, World Scientific, 1994.
- [11] Kapralski, A.: *Supercomputing for solving a class of NP-complete and isomorphic complete problems*, Computer Systems Science and Engineering, 1992, 218–228.
- [12] Knott, G.D.: *A numbering system for binary trees*, Comm. ACM, 1977, 113–115.
- [13] Korsh, J.F.: *Loopless generation of k -ary tree sequences*, Information Processing Letters, 1994, 243–247.
- [14] Krikelis, A., Weems C.C. (Eds): *Associative processing and processors*, IEEE Computer Society Press, 1997.
- [15] Mazurkiewicz T.: *An efficient hardware implementation of a combinations generator*, Technical Sciences 20 (4), 2017, 405–413.
- [16] Mäkinen, E.: *A survey of binary tree codings*. The Computer Journal, 1991, 438–443.
- [17] Pan, Y., Li K.: *Linear array with a reconfigurable pipelined bus system - concepts and applications*, Journal of Information Sciences, 1998, 237–258
- [18] Potter, J.L.: *Associative computing. A programming paradigm for massively parallel computers*, Plenum Press, 1992.
- [19] Potter, J.L., Baker J.W. et al. : *ASC: an associative computing paradigm*, Computer, 1994, 19–25.
- [20] Roelants van Baronaigien D.: *A loopless algorithm for generating binary tree sequences*, Information Processing Letters, 1991, 189–194.
- [21] Roelants van Baronaigien, D., Ruskey F.: *Generating t -ary trees in a -order*, Information Processing Letters, 1988, 205–213.
- [22] Ruskey, F.: *Generating t -ary trees lexicographically*, SIAM Journal of Computing, 1978, 424–439.
- [23] Sedgewick R.: *Permutation generation methods*, ACM Computing Surveys, Vol. 9, Issue 2, 1977, 137–164.
- [24] Skarbek, W.: *Generating ordered trees*, Theoretical Computer Science, 1988, 153–159.
- [25] Skarbek, W.: *On generating all binary trees*, Fundamenta Informaticae, 2007, 505–536.
- [26] Trojanowski, A.E.: *Ranking and listing algorithms for k -ary trees*, SIAM Journal of Computing, 1978, 492–509.

- [27] Vajnovszki, V.: *Constant time algorithm for generating tree Gray codes*, Studies in Informatics and Control, 1996, 15–21.
- [28] Xiang L., Ushijima K., Akl S.G.: *Generating regular k-ary trees efficiently*, The Computer Journal, 2000, 290–300.
- [29] Wang, H., Walker R.A.: *Implementing a scalable ASC processor*, Proc. 17th Int. Parallel and Distributed Processing Symposium, Workshop in Massively Parallel Processing, Nice, France, 2003, 7 pp., DOI: 10.1109/IPDPS.2003.1213482
- [30] Yau, S.S., Fung, H.S.: *Associative processor architecture – a survey*, Computing Surveys, 1977, 3–27.
- [31] Zaks, S.: *Lexicographic generation of ordered trees*, Theoretical Computer Science, 1980, 63–82.
- [32] Zaks, S.: *Generating and ranking t-ary trees*, Information Processing Letters, 1982, 44–48.

C.3. Algorytmy generacyjne w modelu sieci połączeniowej [B11, B23, B39]

W literaturze można spotkać nieliczne prace, w których do generowania obiektów kombinatorycznych używany jest model sieci połączeniowej. W pracy [1] zaproponowano dwa algorytmy generacji losowych permutacji o równomiernym rozkładzie, najlepsze w swojej klasie, dla których punktem wyjścia była losowa sieć połączeniowa generująca kolejne permutacje, symulowana następnie w maszynach PRAM (CRCW i EREW).

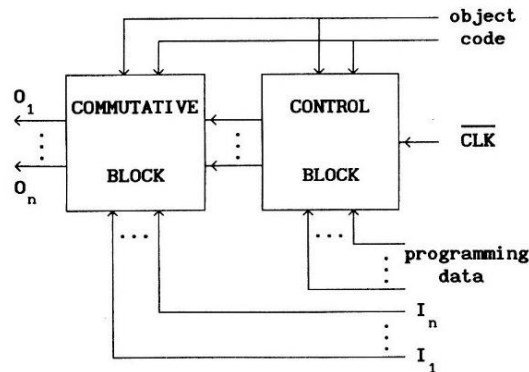
W artykułach [B11, B23, B39], wchodzących w skład cyklu publikacji, wykorzystano model trójkątnej komórkowej sieci połączeniowej (*cellular array*) zbudowanej z komórek dwustanowych (rys. 2) i od nazwisk autorów pracy [3] - W.H. Kautza, K.N. Levitta i A. Waksmana - nazywanej siecią KLV (w wersji lustrzanej – odwrotną siecią KLV). W pracy [5] wykazano, że modelami algebraicznymi obu sieci są iteracyjne dekompozycje symetrycznej grupy permutacji S_n na warstwy lewo- i prawostronne [2]. Autorzy podali algorytmy programowania w czasie $O(n)$ komórek sieci dla uzyskania dowolnej permutacji połączeń wejść i wyjść.



Rys. 2. Trójkątna sieć komórkowa zbudowana z komórek dwustanowych (stany a,b).

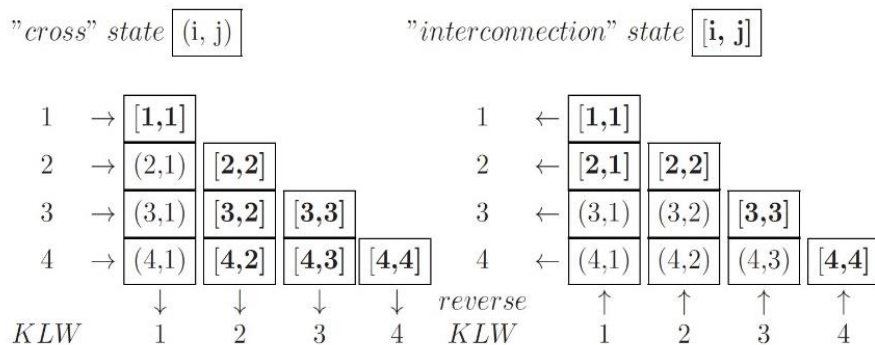
W modelu sieci połączeniowej stan sieci o n wejściach i n wyjściach odpowiada danej konfiguracji kombinatorycznej (np. permutacji wejść sieci otrzymywanej na jej wyjściach). Elementy przełączające (np. komórki), z których zbudowana jest sieć, są sterowane przez odpowiadające im elementy pamięci w układzie sterowania. Liczba sygnałów sterujących komórką zależy od jej możliwości funkcjonalnych (liczby i rodzaju stanów). Generacji sekwencji obiektów kombinatorycznych odpowiada w tym modelu generacja sekwencji stanów układu sterującego

siecią, a tym samym także generacja sekwencji stanów sieci. Zrównoleglenie generacji ma miejsce na poziomie pojedynczego obiektu. Analizując algorytmy generacyjne dla tego modelu należy rozróżnić złożoność czasową programowej generacji obiektów kombinatorycznych od złożoności czasowej generacji sprzętowej, często różnej dla układu sterowania i dla układu komutacyjnego. Podobnie trzeba pamiętać o różnicy pomiędzy asymptotyczną złożonością sprzętową generatora, a jego konkretną realizacją, gdzie rozmiar generatora w systemie komputerowym jest ustalony, a taktowanie układu sterującego jest dostosowane zarówno do czasu propagacji sieci jak i do taktowania układów współpracujących. Struktura generatora w modelu sieci połączeniowej jest pokazana na rys. 3.



Rys. 3. Struktura generatora obiektów kombinatorycznych w modelu sieci połączeniowej [A8].

W pracy [B11] pokazano oryginalną realizację w modelu sieci obliczeniowej algorytmu Zaks'a [6] generującego sekwencję permutacji w oparciu o operację odwracania sufiksu o danym rozmiarze poprzedniej permutacji. Algorytm Zaks'a uogólniono na przypadek odwracania prefiksu, uzyskując dwie różne sekwencje permutacji. Sekwencję rozmiarów sufiksu (prefiksu) wyznacza w pracy [6] formuła rekurencyjna, a w pracy [B11] proces liczenia w odpowiednim kodzie, który może być realizowany przez złożony licznik równoległy. Odwracanie sufiksu (prefiksu) jest realizowane przez sieć połączeniową.



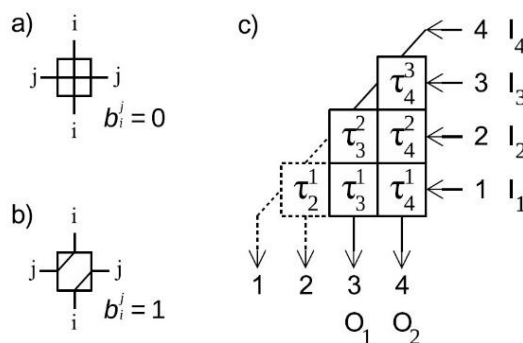
Rys. 4. Sieci komórkowe KLW generujące permutacje [B11].

W odróżnieniu od wcześniejszych algorytmów dla modelu sieci połączeniowej nie programuje się komórek w wierszach lub kolumnach sieci (co najwyżej n komórek), ale jej kolumny lub wiersze odpowiadające za operację odwracania prefiksu (sufiksu). Sieci KLW i odwrotną sieć KLW oraz ich przykładowe zaprogramowanie realizujące odwracanie sufiksu (prefiksu) pokazuje rys. 4.

Algorytm PERMGEN [B11] generuje dwie sekwencje permutacji w nowych porządkach liniowych. Opisano implementację sprzętową generatora permutacji opartego na algorytmie PERMGEN.

W publikacji [B23] podjęto po raz pierwszy w literaturze przedmiotu zadanie generacji permutacji zbioru n -elementowego z dowolnym zabronionym zbiorem F punktów stałych, tj. gdy $\pi(i) \neq i$. Gdy $F \neq \emptyset$, mamy do czynienia z częściowymi nieporządkami zbioru (*partial derangements*). W szczególnych przypadkach: 1) $F = \emptyset$ oznacza permutacje zbioru $\{1, \dots, n\}$; 2) $F = \{1, \dots, n\}$ oznacza całkowite nieporządki (*derangements*) – permutacje bez punktów stałych. Omówiono własności nieporządków zbioru, w prowadzono pojęcie częściowego nieporządku sufiksowego i prefiksowego oraz podano wzory na liczbę nieporządków $PD(n,k)$, gdy $|F|=k$, $k < n$. Algorytmy generacji sformułowano w modelu sieci połączeniowej. Podane zostały warunki, kiedy sekwencja sterująca trójkątnej sieci komórkowej reprezentuje nieporządek oraz zdefiniowano funkcję logiczną wykrywającą stany układu sterowania, a tym samym konfiguracje sieci permutacyjnej, odpowiadające nieporządkom. Wiadomo, że $\lim_{n \rightarrow \infty} \frac{P_n}{D_n} = e$, gdzie: P_n - liczba permutacji zbioru $\{1, \dots, n\}$; D_n - liczba nieporządków zbioru $\{1, \dots, n\}$, a $\lim_{n \rightarrow \infty} \frac{P_n}{PD(n,k)} < e$, gdzie: $PD(n,k)$ – liczba częściowych nieporządków zbioru $\{1, \dots, n\}$. Algorytm generacyjny częściowych nieporządków w modelu sieci połączeniowej różni się od algorytmu generacji permutacji jedynie stałą oszacowania i jest praktycznie użyteczny. O ile wiadomo, jest to pierwszy algorytm generacyjny dla tej klasy obiektów.

W pracy [B39] przedstawiono metodę generowania (n,m) -permutacji. Gdy $n=m$, algorytm VARGEN generuje permutacje zbioru identycznie jak algorytm PG [A7], przy czym podstawą algorytmu generacyjnego jest m -poziomowa iteracyjna dekompozycja grupy S_n na warstwy, $1 \leq m \leq n$. Sekwencja sterująca generowana jest porządku leksykograficznym, podczas gdy sekwencja wynikowa występuje w pewnym porządku liniowym. Druga wersja algorytmu, VARGEN-NR, generuje dokładnie NR obiektów w tym samym porządku jak VARGEN, począwszy od dowolnego zadanego obiektu, co wspomaga generację podzbiorów obiektów w algorytmie równoległym. Podano formułę wyznaczania numeru bieżącego obiektu (*ranking*) oraz algorytm konwersji *liczba porządkowa* \rightarrow *obiekt*, umożliwiający podział zadania generacji zbioru obiektów na podzadania, które mogą być wykonywane równolegle. Analogicznie jak w przypadku generacji permutacji istnieje możliwość implementacji sprzętowej algorytmu. W generacji sprzętowej bierze udział tylko część sieci komórkowej złożona z m kolumn (rys. 5). Sekwencje sterujące generowane są w czasie $O(1)$, podczas gdy sekwencje wynikowe w czasie $O(n)$, odpowiadającym czasowi propagacji sieci.



Rys. 5. Permutacyjna sieć komórkowa generująca (n,m) -permutacje: a) komórka $C[i; j]$ realizująca transpozycję tożsamościową; b) komórka $C[i; j]$ realizująca transpozycję (i,j) ; c) sieć komórkowa C dla $n=4, m=2$. [B39]

W pracy [B39] podano pierwszy iteracyjny algorytm programowania sieci KLV, zastępujący dotychczas stosowany algorytm rekurencyjny [5].

Równoległe algorytmy generacyjne dla modelu komórkowej sieci połączeniowej [B11,B23,B39] są w pełni oryginalnym osiągnięciem habilitanta.

Literatura:

[1] Czumał A., Kanarek P., Kutylowski M., Lorys K.: *Fast generation of random permutations via networks simulation*, *Algorithmica* 21, 1998, 2-20.

[2] Hall M., Paige J.L.: *Complete mapping in finite groups*, *Pacific Journal of Mathematics*, 1955, 541-549.

[3] Kautz W.H., Levitt K.N., Waksman A.: *Cellular interconnection networks*, *IEEE Trans. Computers* 17 (1968), 443-451.

[4] Kulesza K.: *On inverting the VMPC one-way function*, www-old.newton.ac.uk/preprints/NI06009.pdf

[5] Oruç A.Y., Oruç A.M.: *Programming cellular permutation networks through decomposition of symmetric groups*, *IEEE Trans. Computers* 36 (1987), 802-809.

[6] Zaks S.: *A new algorithm for generation of permutations*, *BIT* 24, 1984, 196-204.

C.4. Algorytmy numeracyjne [B13,B26,B30,B39]

Wczesne prace [B1,B2] zawierające pierwsze algorytmy numeracyjne dla kombinacji, nie zostały włączone do cyklu publikacji. W pracy [B1] omówione zostały algorytmy konwersji liczb porządkowych w k -elementowe kombinacje zbioru n -elementowego. W oparciu o przegląd literatury przedstawiono analizę właściwości kilku istniejących algorytmów konwersji (*unranking*), systematyzując wiedzę na temat ich złożoności czasowej i pamięciowej, a następnie poprzez propozycję czterech nowych algorytmów uzupełniono zauważone braki. Różnica pomiędzy różnymi algorytmami wynika ze sposobu wyznaczania współczynników dwumiennych (*restricted factorialing*, programowanie dynamiczne). Z trzech algorytmów o złożoności czasowej $O(n)$ dwa wykorzystują programowanie dynamiczne (UNRANKCOMB-A i UNRANKCOMB-B) i wymagają $O(nk)$ pamięci komputera, a trzeci metodę dziel i zwyciężaj (UNRANKCOMB-D) o takiej samej złożoności czasowej i wymaganych zasobach pamięci tylko $O(k)$. Jest to znaczący postęp w stosunku do algorytmów RANKCINV [2] oraz k th COMBINATION [5] o złożoności czasowej $O(nk)$. Interesujące rozwiązanie stanowi algorytm UNRANKCOMB-C o złożoności czasowej $O(k \log n)$, łączący zalety programowania dynamicznego z wyszukiwaniem binarnym. W pracy [B2] przedstawiono dwa zorientowane sprzętowo równoległe algorytmy unrankingowe dla k -podzbiorów. Jedno z zastosowań wyników pracy [B1] – indeksowanie enumeracyjne – omówiono w [10].

Do cyklu publikacji włączono artykuły [B13,B26,B30,B39] wykorzystujące do numeracji obiektów metodę programowania dynamicznego, obliczenia systoliczne i operacje asocjacyjne. W pracy [B13] omówiono własności drzew t -narnych, ich reprezentacje (z -sekwencje, t -sekwencje, x -sekwencje), przedstawiono pojęcie liczb Ruskey'a [7,9] oraz podano formuły rekurencyjne, opisujące budowę tablic Ruskey'a (RT) dla różnych wartości n i t . Pokazano konstrukcję tablicy RT dla $(n,3)$ -drzew, $n \leq 5$. Następnie przedstawiono algorytm sekwencyjny UNRANKTREE o złożoności $O(nt)$, który przypisuje liczbom porządkowym odpowiadające im drzewa t -narne (porządek leksykograficzny malejący). Udowodniono twierdzenie dotyczące złożoności algorytmu. W algorytmie UNRANKTREE można zrównoleglić:

- 1) tworzenie tablicy współczynników RT (za pomocą obliczeń w dwuwymiarowej tablicy systolicznej);
- 2) poszukiwanie współczynnika w tablicy RT, który reprezentuje odległość pomiędzy dwoma funkcjami wyboru (za pomocą asocjacyjnych operacji wyszukiwania w danym wierszu $\{\leq, maximum\}$).

Otrzymany algorytm UNRANKTREE-PAR działa w czasie $O(n)$.

W publikacji [B26] w oparciu o liczby Ruskey'a, sformułowano nowe formuły rekurencyjne do obliczania współczynników tablic RT i pokazano konstrukcje przykładowych tablic dla $t=2,3,4$. W zakresie algorytmów rankingowych praca zawiera nowy sekwencyjny algorytm RANKTREE o złożoności $O(n)$, poprawiający złożoność $O(nt)$ algorytmów Ruskey'a [9] i Zaks'a [13]. Ponadto wskazano, że obliczenia w pętli algorytmu mogą zostać zrównoleglone i w modelu CRCW PRAM, dzięki operacji współbieżnego zapisu, złożoność algorytmu RANKTREE można zredukować do $O(1)$.

W pracy [B30] omówiono własności i reprezentację w postaci funkcji wyboru co najwyżej n -blokowych podziałów zbioru n -elementowego [4]. Przedstawiono pojęcie liczb Williamson'a oraz podano formuły rekurencyjne, opisujące budowę tablic Williamson'a (WT) dla różnych wartości n i m , gdzie m oznacza liczbę bloków podziału [12]. Pokazano konstrukcję tablicy WT dla (n,m) -podziałów, $1 \leq m \leq n \leq 6$. Następnie przedstawiono algorytm sekwencyjny UNRANKPART o złożoności $O(n^2)$ przypisujący liczbom porządkowym (porządek leksykograficzny) odpowiadające im podziały co najwyżej n -blokowe. Podano twierdzenie dotyczące złożoności algorytmu. W algorytmie UNRANKPART można zrównoleglić: 1) tworzenie tablicy współczynników WT (dwuwymiarowa tablica systoliczna); 2) poszukiwanie współczynnika w tablicy WT za pomocą asocjacyjnych operacji wyszukiwania w wierszu $\{\leq, maximum\}$. Otrzymany algorytm UNRANKPART-PAR działa w czasie $O(n)$. Bezpętlowe algorytmy numeracyjne o złożoności $O(kn^2)$ dla drzew t -narnych reprezentowanych przez z -sekwencje w porządku Gray'a zostały opublikowane w [1].

W pracy [B39] podano formułę wyznaczającą liczbę porządkową (n,m) -permutacji w porządku leksykograficznym dla sekwencji sterującej w czasie $O(m)$. Algorytm UNRANKVAR, wykorzystujący przygotowaną wcześniej tablicę jednowymiarową współczynników W posiada złożoność $O(m)$, co wykazuje odpowiednie twierdzenie (por. [3]). W pracach A. Kaprałskiego [5,6] można znaleźć szereg analogicznych algorytmów programowania dynamicznego dla obiektów kombinatorycznych, ale stosowane tablice są budowane w kolejności od największych do najmniejszych indeksów tablic, co powoduje, że w przypadku zwiększenia rozmiaru problemu, nie wystarczy rozbudować tablice, ale trzeba je konstruować od nowa. Wady takiego rozwiązania są oczywiste.

Jak już wspomniano wcześniej, algorytmy konwersji *liczba porządkowa* \rightarrow *obiekt kombinatoryczny* znajdują zastosowanie w równoległej generacji zbioru obiektów w wyszukiwaniu wyczerpującym, w generacji losowej stosowanej m.in. testowaniu oprogramowania, algorytmach randomizowanych [8]. Innym zastosowaniem algorytmów rankingowych jest kodowanie enumeracyjne, w którym wybrane klasy równie prawdopodobnych wiadomości/komunikatów (*message*) są konwertowane do klas obiektów kombinatorycznych, zapewniając unikalne indeksowanie obiektów wewnątrz klasy [10]. Przyporządkowany indeks i kod klasy enumeracyjnej reprezentują skompresowane dane wiadomości. W metaheurystykach opartych na populacjach rozwiązań dopuszczalnych (np. w algorytmach genetycznych [11]) algorytmy numeracyjne mogą być zastosowane do losowej generacji populacji i w operacji krzyżowania obiektów kombinatorycznych (chromosomów).

Literatura:

- [1] Ahmadi-Adl A., Nowzari-Dalini A., Ahrabian, H.: *Ranking and unranking algorithms for loopless generation of t -ary trees*, Logic Journal of the IGPL 19 (2011), 33-43, DOI: 10.1093/jigpal/jzp097
- [2] Akl S.G.: *Design and analysis of parallel algorithms*, Prentice Hall, Englewood Cliffs, N.J., 1989, 148-150
- [3] Dobrev P.G., Minov Y.O., Trifonov V.T.: *Encoding and decoding of variations without repetitions*, Cybern Syst Anal 32 (1996) 741-744. DOI: 10.1007/BF02367777
- [4] Hutchinson G.: *Partitioning algorithms for finite sets*, Comm. ACM, Vol. 6, 1963, 613-614
- [5] Kaprałski A.: *New methods for generation permutations, combinations and other combinatorial objects in parallel*, J. Parallel and Distrib. Computing, Vol. 17, 1993, 315-326.

- [6] Kapralski, A.: *Modelling arbitrary sets of combinatorial objects and their sequential and parallel generation*. *Studia Informatica*, Silesian University of Technology (monografia), 2000.
- [7] Knott, G.D.: *A numbering system for binary trees*, *Comm. ACM*, 1977, 113–115.
- [8] Motwani R., Raghavan P.: *Randomized algorithms*, Cambridge University Press 1995.
- [9] Ruskey F.: *Generating t -ary trees lexicographically*, *SIAM Journal of Computing*, Vol. 7, 1978, 424–439.
- [10] Tomič R.V.: *Quantized indexing: background information*, 1stWorks Corporation Technical Report TR05-0625, 1st Works Corporation, June 25, 2005, 39 pp. (www.1stWorks.com).
- [11] Üçoluk G.: *A method for chromosome handling of r -permutations of n -element set in genetic algorithms*, *IEEE Int. Conf. on Evolutionary Computations, ICEC'97*, Nagoya, Japan, 1–4.
- [12] Williamson S.G.: *Ranking algorithms for lists of partitions*, *SIAM J. of Computing*, Vol 5., No. 4, 1976, 602–617.
- [13] Zaks S.: *Generating and ranking of t -ary trees*, *Information Processing Letters*, 1984, 44–88.

C.5. Generacja populacyjna [B15, B17, B18]

Chronologicznie pierwszą pracą autora związaną z generacją populacyjną była publikacja [B7]. Wykazano w niej, że reprezentacja permutacji, stosowana w algorytmie generacyjnym [A7,A8] jako sekwencja sterująca (reprezentacja Coset), pod działaniem zwykłego operatora krzyżowania daje zawsze legalne potomstwo i wraz z jednopunktową mutacją może znaleźć zastosowanie w algorytmach ewolucyjnych dla problemów permutacyjnych. Inną znaną reprezentacją permutacji o takiej własności jest reprezentacja porządkowa (Ordinal) stosowana dla Problemu TSP (*Travelling Sales Person*) [8]. W reprezentacji tej, trasy komiwojażera są normalizowane przez listę referencyjną elementów permutacji, co również umożliwia stosowanie klasycznego operatora krzyżowania.

W pracy [B14,B15] do klasycznego problemu korowania grafu (GCP) [7] wykorzystano równoległy algorytm ewolucyjny w modelu migracyjnym z wieloma populacjami (model wyspowy) [2,4]. Oprócz klasycznych operatorów krzyżowania, mutacji i selekcji osobników, tworzących stochastyczny mechanizm generowania kolejnych populacji algorytmu ewolucyjnego, specyficznymi właściwościami algorytmu równoległego są: ilość i rozmiar koewoluujących podpopulacji, parametry migracji (charakterystyka, liczebność i częstotliwość okresowej migracji) oraz sposób zrównoleglenia obliczeń [4].

Wstępne wyniki badań przeprowadzonych przy pomocy programu *PGA_for_GCP* przedstawione zostały w [B14], a znacznie bardziej szczegółowe w [B15]. W badaniach wykorzystano benchmarki w postaci sześciu grafów z repozytorium DIMACS (*anna, david, huck, miles500, myciel7, mulsol.i.1*) o znanych liczbach chromatycznych $\chi(G)$ [6,10-12]. Zaprojektowano nowe elementy algorytmu genetycznego w postaci dedykowanych operatorów rekombinacji: krzyżowania uniwersalnego dla problemów podziałowych SPPX (*Sum-Product Partition Crossover*) oraz dedykowanego dla problemów kolorowania grafu CEX (*Conflict Elimination Crossover*) oraz funkcji kosztu dla operatora selekcji proporcjonalnej (koło ruletki). Wykazano, że algorytm ewolucyjny, w zależności od rozmiaru i trudności problemu oraz założonego warunku stopu, może stanowić narzędzie do dokładnego bądź przybliżonego rozwiązywania problemu GCP. Badanie algorytmu równoległego przeprowadzono na maszynie sekwencyjnej drogą symulacji. Zauważono, że zrównoleglenie obliczeń w modelu migracyjnym może nie tylko skrócić czas obliczeń, ale również skutkować poprawą jakości rozwiązania, co zawdzięczamy okresowej migracji najlepszych osobników pomiędzy podpopulacjami. Parametry algorytmów i operatorów zostały dostrojone empirycznie – omówiono wpływ tych ustawień na efektywność algorytmu. Bardziej precyzyjne dostrojenie parametrów wymaga wyznaczenia tzw. map ciepła (*heat maps*), ilustrujących w postaci gradientu kolorów średnią wartość funkcji dopasowania albo czas osiągnięcia zbieżności [3]. Przeprowadzono badania porównawcze operatorów mutacji, krzyżowania oraz różnych konfiguracji operatorów krzyżowania i mutacji. Najlepsza mutacja First Fit (FF) działa szczególnie dobrze

z krzyżowaniem CEX, które okazało się operatorem krzyżowania lepszym niż SPPX, UISX i GPX ze względu na kompromis czasu obliczeń i jakości rozwiązania. Do opisanych badań nie został jeszcze włączony efektywny operator krzyżowania BCX, opisany w późniejszej pracy [9]. Praca [B15] została omówiona w pierwszej w literaturze światowej książce poświęconej w całości równoległym metaheurystykom [1], jako przykład pierwszego zastosowania metaheurystyki równoległej do problemu kolorowania grafu [5].

Rozszerzone i pogłębione wyniki dotyczące zastosowań algorytmów ewolucyjnych do klasycznego problemu kolorowania wierzchołków przedstawiono w pracy [B17]. Poszerzono kontekst pracy, zakres badań i bazę badanych grafów. Pogłębiono analizę poszczególnych składników algorytmu ewolucyjnego. Podano przykłady i rysunki ilustrujące działanie operatorów. Porównanie algorytmu sekwencyjnego i równoległego pokazano na przykładzie kolorowania grafu *queen8.8* z $\chi(G)=9$. Do badania ewolucji na trzech izolowanych wyspach (bez migracji) wykorzystano grafy *r.50* (graf losowy o gęstości krawędzi 50%) i graf *le450.15b* oraz operatory GPX i CEX. Algorytm równoległy charakteryzował się większą zbieżnością w kierunku optymalnego rozwiązania niż algorytm sekwencyjny i lepiej unikał lokalnych minimów. Algorytm z operatorem CEX osiągał przy tej samej liczbie iteracji porównywalne rozwiązania jak algorytm z operatorem GPX w czasie ok. 10-krotnie krótszym. Dla tych samych grafów oraz czterech operatorów krzyżowania i dwóch mutacji badano optymalny odstęp pomiędzy migracjami w modelu wyspowym, uzyskując w zależności od badanego grafu optymalne wartości w zakresie 2-10 i potwierdzenie efektywności mutacji FF. Badając na grafach *games250*, *myciel7* i *mulsol.i.4* skład migracji wykazano, że najbardziej efektywna jest migracja najlepszych osobników, chociaż dla dużych grafów takich jak *mulsol.i.4* migracja losowa z operatorem CEX może okazać się dobrym rozwiązaniem. Dla pięciu grafów DIMACS przeprowadzono badanie czterech operatorów krzyżowania ustalając ich kolejność ze względu na kryterium jakości rozwiązania (1-2.CEX/GPX, 3. UISX, 4. SPPX) i czasu obliczeń (1.CEX, 2-3.GPX/SPPX, 4.UISX).

W pracy [B18] przeprowadzono porównanie dwóch uniwersalnych reprezentacji permutacji dla problemu TSP. Instancjami problemu TSP były losowo wygenerowane grafy ważone z $N=100$, 200 i 500 wierzchołkami. Wśród parametrów programu implementującego algorytm ewolucyjny znalazł się schemat kodowania (Coset, Ordinal, Path) wraz z ośmioma kombinacjami czterech krzyżowań (OX, CX, PMX, heurystyka Grafenstette) i dwóch mutacji (Inversion, Transposition) [B7,8]. Przeprowadzono badania porównawcze dla różnych reprezentacji permutacji i różnych konfiguracji operatorów. Stwierdzono, że standardowe operatory rekombinacji pracowały lepiej z reprezentacją Coset niż Ordinal, ale obie reprezentacje ustępowały reprezentacjom dedykowanym (Path).

W pracy [B18] porównano również reprezentacje i operatory dla problemów podziałowych (SPPX, CEX, GPX, Standard, UISX). Jako przykład referencyjny posłużył problem kolorowania krawędzi grafów *queen5.5* i *myciel7* z repozytorium DIMACS. Uniwersalny operator SPPX dla reprezentacji blokowej podziałów okazał się lepszy od standardowego operatora krzyżowania w reprezentacji wektorowej (assignment). Dorównywał dedykowanym operatorom w kolorowaniu bezkonfliktowym ale ustępował im pod względem jakości rozwiązania.

Badania nad kolorowaniem grafów algorytmem PEA podjęto również w równoległym modelu dyfuzyjnym [B32] podając wnioski dotyczące optymalnego typu i rozmiaru sieci typu mesh, liczebności podpopulacji w węzłach sieci itp. Wykorzystano m.in. wprowadzone w poprzednich pracach efektywne operatory krzyżowania.

Literatura:

- [1] Alba, E. (Ed.): *Parallel metaheuristic – a new class of algorithms*, John Wiley & Sons, 2005.
- [2] Bäck, T.: *Evolutionary algorithms in theory and practice*, Oxford University Press, 1996.
- [3] Błażej P., Wnętrzak M., Mackiewicz P.: *The role of crossover operator in evolutionary-based approach to problem of genetic code optimization*, BioSystems 150, 2016, 61–72, DOI: 10.1016/j.biosystems.2016.08.008

- [4] Cantú-Paz, E.: *Efficient and accurate parallel genetic algorithms*, Kluwer, 2000.
- [5] Crainic T.G., Nourredine, H.: *Parallel meta-heuristics applications*, [in:] Alba E. (Ed.): *Parallel metaheuristics: a new class of algorithms*, John Wiley & Sons, 2005.
- [6] Johnson, D. S., Trick, M. A.: *Cliques, coloring and satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discr. Math. and Theor. Comp. Sc. Vol. 26, 1996.
- [7] Kubale, M. (Ed.): *Optymalizacja dyskretna. Modele i metody kolorowania grafów*, WNT, Warszawa, 2002.
- [8] Michalewicz Z.: *Evolutionary Algorithms + Data Structures = Evolutionary Programs*, 3rd ed., Springer, 1996.
- [9] Myszkowski, P.B.: *Solving scheduling problems by evolutionary algorithms for graph coloring problem*, [in:] Xhafa F., Abraham A.: *Metaheuristics for scheduling in industrial and manufacturing applications*, Studies in Computational Intelligence, Vol. 128, 2008, 145–167.
- [10] COLOR web site. Available at: <http://mat.gsia.cmu.edu/COLOR/instances.html> .
- [11] DIMACS ftp site. Available at: <ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/> .
- [12] COLORING3 web site. Available at: <http://mat.gsia.cmu.edu/COLORING03>

C.6. Podsumowanie

W przedstawionym cyklu publikacji uzyskano wartościowe i oryginalne wyniki, które są cytowane w światowej literaturze naukowej przez innych autorów. Z wyjątkiem prac 4 i 5 pozostałe publikacje cyklu są samodzielnym dorobkiem habilitanta. Do osiągnięć zawartych w cyklu publikacji należą:

- **nowe algorytmy generacyjne w modelu sieci połączeniowej dla n -permutacji [B11] i (n,m) -permutacji B[39] (nowe porządki generacji)**
- **nowe algorytmy generacyjne w modelu sieci połączeniowej dla częściowych nieporządków [B23] (nowa klasa obiektów, w której permutacje i nieporządki są przypadkami szczególnymi; nowe reprezentacje i porządki generacji)**
- **nowe algorytmy generacyjne w modelu asocjacyjnym ASC SIMD dla drzew binarnych i t -narnych [B8,B26] (nowa reprezentacja obiektów w postaci t -sekwencji, pierwszy algorytm dla reprezentacji binarnej w postaci x -sekwencji)**
- **nowy algorytm równoległy w modelu złożonym (ASC SIMD + licznik równoległy) dla dokładnie m -blokowych podziałów zbioru n -elementowego [B19]**
(powyższe algorytmy mogą znaleźć zastosowanie do równoległej sprzętowej generacji obiektów kombinatorycznych w układach ASIC/FPGA)
- **nowe algorytmy numeracyjne i wskazano ich zastosowania [B13,B30,B39] (technika programowania dynamicznego i inne).**
- **nowe operatory krzyżowania w obszarze generacji populacyjnej [B7,B15,B18] (projekt, testowanie, zastosowania)**
- **pierwsza w literaturze równoległa metaheurystyka do wierzchołkowego kolorowania grafów [B15,B16] (równoległy algorytm ewolucyjny).**

Według dostępnych baz danych publikacje habilitanta na temat algorytmów generacji i numeracji obiektów wraz z zastosowaniami były łącznie cytowane w literaturze światowej ponad sto razy, m.in. przez takich autorów jak S.G. Akl, I. Stojmenović, F. Ruskey, T. Sasao, T.G. Crainic, F. Glover, J.-K. Hao i innych.

Pozostałe osiągnięcia naukowe

5. Tematyka badań naukowych poza jednorodnym cyklem publikacji

D. Przetwarzanie równoległe

D.1. Procesor asocjacyjny z wieloma komparandami i wieloma operacjami wyszukiwania oraz dedykowane algorytmy przetwarzania problemów kombinatorycznych [B5, B8, B12, B25]

W pamięciach i procesorach asocjacyjnych (adresowanych zawartością) [4,6,7,11,12,13] równoległe wyszukiwanie rekordów będących w określonej relacji z wzorcem, np. $\{=, \neq\}$, może dotyczyć tylko wybranych pól i podzbioru rekordów, a w przypadku przetwarzania macierzy binarnych – wybranych kolumn i wierszy (*slice*, *word*), co uzyskuje się przez maskowanie. W wyszukiwaniu wyczerpującym zachodzi potrzeba generacji określonych obiektów kombinatorycznych, wykorzystywanych w postaci binarnej jako maski dla kolumn i/lub wierszy (kombinacje, bloki podziałów) oraz połączenia wzorca z tablicą danych (permutacje) [6]. W [A7] opisano w zarysie podstawy teoretyczne i budowę programowanego generatora permutacji, kombinacji i podziałów oraz jego zastosowanie jako komponentu w równoległym przetwarzaniu asocjacyjnym. Artykuł bazuje na pracy doktorskiej [A7].

W klasycznym przetwarzaniu asocjacyjnym (SIMD) z pojedynczym wzorcem (*pattern*, *comparand*) i zbiorem danych, złożoność operacji algebry relacyjnej $\{=, \neq, <, >, \leq, \geq\}$ nie zależy od rozmiaru zbioru danych. Uzyskiwane przyspieszenie zależy od trybu pracy procesora i wiąże się z kosztem sprzętu niezbędnego do zrównoleglenia operacji porównania we wszystkich kombinacjach trybów: *bit-serial/bit-parallel word-serial/word-parallel* [10]. W trybie *bit-parallel word-parallel* wykonanie operacji porównania z wzorcem odbywa się w czasie $O(1)$, niezależnym od rozmiaru zbioru n i długości słowa p [4]. Digby [1] zaproponował architekturę asocjacyjną z wieloma komparandami (*multiassociative*), w której relacje pomiędzy elementami zbiorów komparandów i zbioru danych nie zależą od liczebności tych zbiorów, a jedynie od długości słowa p , która jest stała. Porównania nm par wzorzec-dane odbywają się równoległe, a ich wyniki zapisywane są w tablicy tagów TA, której zawartość podlega dalszemu przetwarzaniu asocjacyjnemu z jednym komparandem drugiego poziomu. Wynik tego przetwarzania jest zapisywany w wektorze – tagu drugiego poziomu T2. Dodatkowo dla każdego wzorca można badać inną relację ze zbiorem danych (każdorazowo programowaną). W pracy [1] zdefiniowano 24 operacje logiczne i 10 arytmetycznych. Idea D. Digby’ego spotkała się z ironiczną uwagą C.C. Fostera, który w swojej książce [4] zanegował możliwość budowy tak skomplikowanego procesora ze względu na złożoność sprzętową pojedynczej komórki, rosnącą wykładniczo z liczbą implementowanych operacji. Jednak warto zauważyć, że w bardzo wielu zastosowaniach nie są wymagane wszystkie możliwe operacje, a jedynie pewien ich podzbiór. Ponadto relacje tworzą pary (np. $\{=, \neq\}$, $\{>, \leq\}$, $\{\geq, <\}$). Aby uzyskać wartość logiczną jednego porównania, wystarczy zanegować wynik porównania przeciwnego, co zmniejsza złożoność sprzętową komórki procesora.

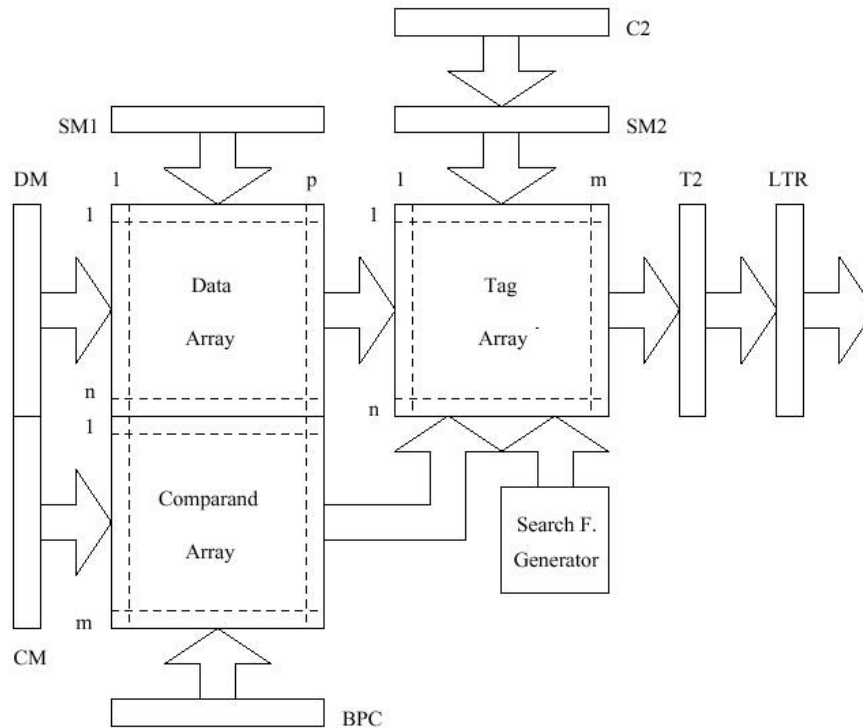
W pracy [B5] przedstawiono algorytmy asocjacyjne dla problemów należących do różnych klas złożoności obliczeniowej: SET INCLUSION, GEOMETRIC RANGE SEARCH (dwie wersje), MATRIX INCLUSION, NEIGHBOURHOOD, MULTIPLE SEARCH, MULTI-LIST RANKING, wykorzystujące ograniczony do sześciu zbiorów operacji asocjacyjnych. Algorytmy te składają się z pięciu etapów:

- 1) wczytania danych (DA, CA);
- 2) zaprogramowania operacji procesora (maski DM, CM, SM1, SM2, wzorzec C2, rodzaje wyszukiwania w kolumnach TA);
- 3) przetwarzania asocjacyjnego pierwszego poziomu (BPC, TA);

- 4) przetwarzania asocjacyjnego drugiego poziomu (T2);
- 5) wyprowadzenia wyniku (LTR).

Przestawione algorytmy asocjacyjne wykorzystują równoległe operacje porównania elementów zbioru danych i zbioru operandów, przyspieszając obliczenia o czynnik nm .

Biorąc pod uwagę stały postęp technologii FPGA, zdecydowano się zweryfikować ideę procesora Digby'ego poprzez pokazanie jego użyteczności w rzeczywistych zastosowaniach. Pierwsza implementacja procesora z wieloma komparandami, pokazanego na rys. 6 została oparta na bazie opisu strukturalnego (schematic) [B8]. Wybrano 16 operacji logicznych dla komórek TM i określono ich początkowe programowanie. Do implementacji sprzętowej wybrano sześć podstawowych operacji logicznych (relacji).



Rys. 6. Schemat blokowy procesora asocjacyjnego z wieloma komparandami [B5,B8].

W drugiej chronologicznie implementacji o analogicznej funkcjonalności [B25] użyto języka opisu sprzętu VHDL. Dla danego zbioru relacji określony został maksymalny rozmiar procesora w danym układzie FPGA oraz maksymalna częstotliwość jego taktowania. Układy zostały zaprojektowane w środowisku XILINX Foundation ISE i przetestowane w symulatorze ModelSim. Poprawność pracy procesora zbadano także na wybranych przykładach (CLIQUE, INDEPENDENT SET, SET INCLUSION) w układzie docelowym FPGA. W układach Spartan XC3S200 uzyskano maksymalny rozmiar procesora $n=m=p=18$ przy taktowaniu 90,0 MHz. W układach Virtex XC4VLX15 procesor o takim samym rozmiarze mógł pracować z częstotliwością 198,0 MHz. W tym przypadku 324 równoczesne porównania były wykonywane w czasie 5.05 ns/bit. Maksymalny rozmiar procesora w układzie Virtex wynosił $n=m=p=34$, przy taktowaniu 145,6 MHz. Dla porównania, pierwsza implementacja procesora [B8] wykonana w roku 2002 w układzie FPGA XC4005XL, osiągnęła rozmiar procesora $n=m=p=5$, przy częstotliwości zegara 18.7 MHz. Szczegółowe dane zawierają cytowane artykuły. Po raz pierwszy w literaturze przedmiotu wykazano empirycznie, że budowa procesora Digby'ego dla określonego podzbioru relacji zależnego od zastosowania, jest praktycznie osiągalna. W pracy [B25] opisano dodatkowo symulator procesora, który jest wykorzystywany jako narzędzie dydaktyczne ilustrujące pracę procesora SIMD.

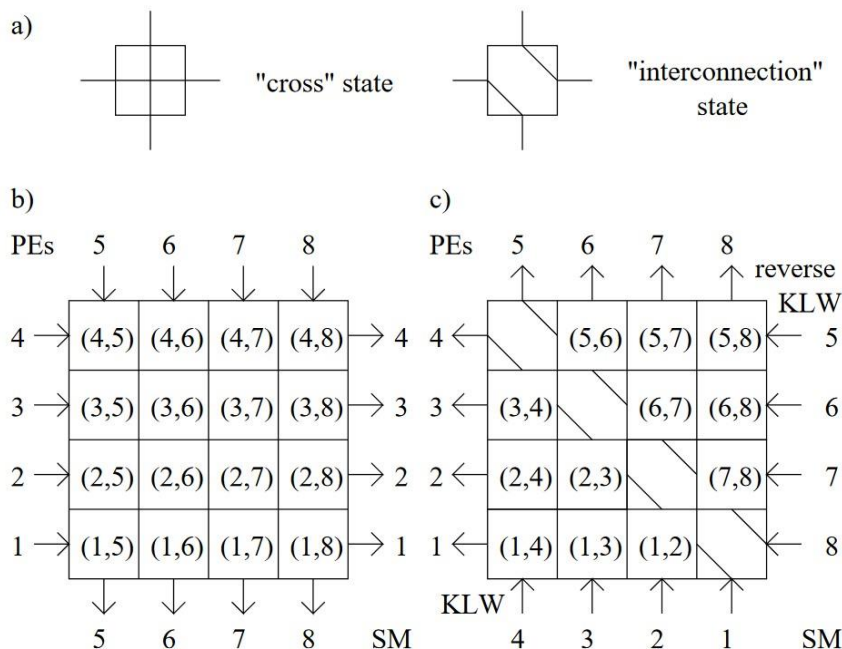
Badania nad zastosowaniami procesora objęły zarówno przetwarzanie równoległe klasy problemów kombinatorycznych posiadających reprezentację w postaci macierzy binarnej jak i algorytmy grafowe (akceleracja obliczeń algorytmów wielomianowych). Te ostatnie algorytmy zostały opracowane w Centrum Obliczeniowym w Nowosybirsku dla modelu maszyny równoległej STAR. W wyniku korespondencyjnej współpracy nad sprzętową implementacją wybranych operacji algebry relacji, powstała koncepcja asocjacyjnego procesora dla problemów grafowych AGP przedstawiona w pracy [B12], w której podano również procedury WCOPY, WMERGE, TMERGE, HIT, MATCH, SETMIN, SETMAX, o niskiej złożoności czasowej, wykorzystane w asocjacyjnych algorytmach grafowych [8,9,10]. Dalsze prace w tym obszarze badać kontynuuje dr A.S. Nepomniaschaya, Supercomputer Software Department, Institute of Computational Mathematics and Mathematical Geophysics, RAS, Siberian Branch.

Literatura:

- [1] Digby D.: *A search memory for many-to-many comparisons*, IEEE Trans. on Computers, C-22: 768–772, 1973.
- [2] Garey, R., Johnson, D. S.: *Computers and intractability. A guide to the theory of NP-completeness*, Freeman, 1979.
- [3] Greenlaw R., Hoover H.J. : *Limits to parallel computations : P-completeness theory*, Oxford University Press 1995
- [4] Foster C.C.: *Content addressable parallel processors*, Van Nostrand Reinhold, 1976.
- [5] Kapralski A. : *Supercomputing for solving a class of NP-complete and isomorphic complete problems*, Computer Systems Science and Engineering Vol.7 (1992) No.4, 218–228
- [6] A. Kapralski A.: *Sequential and parallel processing in depth search machines*, World Scientific, 1994
- [7] Krikelis A., Weems C. (Eds.): *Associative processing and processors*, IEEE Computer Society Press, 1997.
- [8] Nepomniaschaya A.: *Associative Parallel Algorithm for Dynamic Reconstruction of a Minimum Spanning Tree After Deletion of a Vertex*. Proc. PaCT'2005 (Parallel Computing Technologies), Lect. Notes in Comp. Sci. 3606 (2005), 159–173.
- [9] Nepomniaschaya A.: *Efficient Implementation of the Italiano Algorithms for Updating the Transitive Closure on Associative Parallel Processors*, Fundamenta Informaticae 89 (2-3), 313–329 (2008)
- [10] Nepomniaschaya A.: *Efficient Parallel Implementation of the Ramalingam Decremental Algorithm for Updating the Shortest Paths Subgraph*, Computing and Informatics 32(2), (2013), 331–354
- [11] Potter J.L. (1992) *Associative computing. A programming paradigm for massively parallel computers*, Plenum Press 1992.
- [12] Potter, J.L., Baker J.W. *et all.* : *ASC: an associative computing paradigm*, Computer, 1994, 19–25.
- [13] Yau S.S., Fung H.S.: *Associative processor architecture – a survey*, Computing Surveys 9 (1977), 3–27

D.2. Kwadratowa sieć połączeniowa [B9]

W artykule [B9] wykazano, że kwadratowa sieć komórkowa składająca się z $(n^2/4)$ komórek (czwórników) jest wystarczająca do realizacji dowolnej permutacji π w dwóch przejściach (rys. 7). Sformułowano algorytm DATAPERMUTATION o złożoności $O(n)$ do programowania takiej sieci. Algorytm wykorzystuje jako swoje komponenty algorytm SQUAREFACTOR do programowania sieci w pierwszym przejściu oraz dwa znane algorytmy $O(n)$ dla trójkątnych sieci komórkowych: KLWFACTOR i REVKLWFACTOR [2] do programowania sieci w drugim przejściu. Alternatywę dla tych ostatnich algorytmów rekurencyjnych stanowią algorytmy iteracyjne [B39].



Rys. 7. Kwadratowa sieć komórkowa (n=8): a) stany komórki 2x2 (czwórnika); b) etykietowanie kwadratowej tablicy komórek w pierwszym przejściu; c) etykietowanie kwadratowej tablicy komórek w drugim przejściu [B9].

Kwadratowa sieć połączeniowa zapewnia efektywną dwufazową komunikację pomiędzy procesorami w modelu SIMD za pomocą pamięci dzielonej (*shared memory*) [1]. Nie posiada długich połączeń ani skrzyżowań połączeń pomiędzy komórkami. Ze względu na regularną strukturę dobrze nadaje się do implementacji w układzie VLSI.

Literatura:

[1] Lenoski D.E., Weber W.-D.: Scalable shared memory multiprocessing, Morgan Kaufman Publishers, Inc. 1995.

[2] Oruç A.Y., Oruç A.M.: *Programming cellular permutation networks through decomposition of symmetric groups*, IEEE Trans. Computers 36 (1987), 802-809. DOI: 10.1109/TC.1987.1676977

E. Równoległe metaheurystyki

E.1. Sumacyjne kolorowanie grafów [B20,B29]

Celem pracy [B20], poświęconej problemowi sumacyjnemu kolorowaniu grafów (MSCP) [12,16], było zastosowanie równoległego algorytmu ewolucyjnego (PEA) do przybliżonego wyznaczania sum chromatycznych $\sum(G)$ dla grafów DIMACS. W pracy wykorzystano program *PGA_for_GCP* rozbudowany o nowe funkcjonalności (redefinicja funkcji kosztu). W badaniach eksperymentalnych wykorzystano obie reprezentacje grafów: blokową i wektorową, operatory krzyżowania CEX i GPX oraz mutację First Fit selekcję proporcjonalną.

Praca [B20] przyniosła ponadto nowe wyniki teoretyczne dotyczące oszacowania minimalnej sumy chromatycznej grafu $\sum(G)$, definiowanej jako suma kolorów bezkonfliktowego pokolorowania wierzchołków grafu G , oraz liczby sumy chromatycznej $s(G) \geq \chi(G)$, czyli minimalnej liczby kolorów w optymalnym kolorowaniu sumacyjnym. Publikacja [B20] zapoczątkowała w środowisku naukowym rozległe badania grafów z repozytorium DIMACS pod kątem wyznaczenia najlepszych oszacowań górnych i dolnych ich sum $\sum(G)$, które generalnie nie są znane. Wyniki teoretyczne pracy

zostały ujęte w twierdzeniu formułującym dwa nowe, w kontekście znanej literatury przedmiotu, oszacowania dolne na sumę chromatyczną :

- 1) $n + s(G)(s(G)-1)/2 \leq \sum(G)$
- 2) $n + \chi(G) (\chi(G)-1)/2 \leq \sum(G)$

O ile dla niektórych grafów liczba $\chi(G)$ jest znana, to znajomość $s(G)$ wiąże się ze znajomością optymalnego kolorowania sumacyjnego o minimalnej liczbie kolorów.

W pracy podano nowe oszacowania dolne $\sum(G)$ wynikające z nierówności 2) dla 15 z 16 grafów DIMACS wybranych do badań. Dla wszystkich 16 grafów znacznie poprawiono eksperymentalnie znane teoretycznie oszacowania górne $s(G)$ (w dwóch przypadkach oszacowanie zostało wyrównane). W 14 przypadkach na 16, z wyjątkiem grafów *queen6.6* i *queen8.8*, uzyskano wynik $s(G)=\chi(G)$. W podsumowaniu artykułu wyrażono przekonanie, że uzyskane wyniki będą stanowić odniesienie dla dalszych badań, których celem będzie wyznaczenie sum chromatycznych $\sum(G)$ dla wszystkich trudnych do pokolorowania grafów z repozytorium DIMACS, co byłoby dopełnieniem ich dotychczasowej charakterystyki.

Zadanie to zostało podjęte, ukazało się wiele artykułów przynoszących nowe wyniki uzyskane różnymi metodami [1,2-7,8,9-11,14-15,17-18,19,20], a praca [B20] została zacytowana ponad 30 razy. Stan badań nad charakterystyką sumacyjną grafów DIMACS podsumowano m.in. w [B29,11,15]. Uzyskane wyniki dowodzą znaczącego postępu w charakterystyce chromatycznej grafów DIMACS. Dla wielu grafów zmniejszyła się ulepszana drogą kolejnych przybliżeń miara zdefiniowana w pracy [B29]:

$$gap = [(min(UB_{th}, UB_{exp}) - max(LB_{th}, LB_{exp})) * 100\% / min(UB_{th}, UB_{exp})],$$

gdzie: UB_{th}, UB_{exp} – ograniczenia górne, wyznaczone teoretycznie i doświadczalnie; LB_{th}, LB_{exp} – ograniczenia dolne, wyznaczone teoretycznie i doświadczalnie. W zestawieniu podanym w artykule [B29] w 25 przypadkach na 26 stwierdzono równość $s(G)=\chi(G)$, a w jednym przypadku wartość $s(G)$ zawiera się w zbiorze $\{\chi(G), \chi(G)+1\}$, pomimo faktu iż grafy DIMACS nie są grafami trywialnymi.

W zestawieniu [11] podano nowe oszacowania $\sum(G)$ dla 94 grafów, w tym oszacowania dokładne dla 21 grafów. Algebraiczne oszacowania dolne sumy chromatycznej $\sum(G)$ z pracy [B20] zostały w latach 2016-2017 poprawione przez nowe oszacowania podane w pracach [10,15] bazujące w pierwszym przypadku na podziale grafu na kliki a w drugim, nazwanym LBM_{\sum} , na wyznaczeniu maksymalnego stabilnego zbioru grafu G przez wyznaczenie maksymalnej kliki grafu dopełniającego algorytmem dokładnym IncMaxCLQ [13]. W kilku przypadkach, np. dla grafów *flat1000-50-0*, *flat300-20-0*, *le450-5c*, *le4505d*, znaleziono rozwiązania dokładne [15]. W najnowszej pracy [21] przedstawione zostały nowe techniki zwane *extraction and backward expansion search* (EBES) do wyznaczania górnych i dolnych oszacowań dla 35 dużych instancji problemu MSCP, które umożliwiły poprawienie 31 znanych dotąd najlepszych oszacowań (19 górnych i 12 dolnych).

Literatura:

- [1] Bouziri, H., Jouini, M.: *A tabu search approach for the sum coloring problem*, Electronic Notes in Discrete Mathematics, Vol. 36, 2012, 915–922, DOI: 10.1016/j.endm.2010.05.116
- [2] Douiri M., El Bernoussi S.: *New algorithm for the sum coloring problem*, Int. J. Contemp. Math. Sciences, Vol. 6, No.10, 2011, 453 – 463.
- [3] Douiri M., El Bernoussi S.: *A new heuristic for the sum coloring problem*, Applied Math. Sciences, Vol. 5, No.63, 2011, 3121–3129.
- [4] Douiri M., El Bernoussi S.: *A hybrid algorithm for the sum coloring problem*, Proc. 2011 International Conference on Multimedia Computing and Systems (ICMCS), 7–9 April 2011, Ouarzazate, Morocco, 1–4.

- [5] Douiri S.M., El Bernoussi S.: *An ant colony optimization for the sum coloring problem*, International Journal of Applied Mathematics & Statistics, Vol. 27, No.3, 2011, 102–110.
- [6] Douiri S.M., El Bernoussi S.: *A new ant colony optimization algorithm for the lower bound of sum coloring problem*, J. Math. Modeling and Algorithms Vol.11, No.2., 2012, 181–192.
DOI: 10.1007/s10852-012-9172-x
- [7] Douiri S.M., El Bernoussi S.: *Lower Bound of the Sum Coloring Problem in Graph*, Proceeding of the International Symposium on Combinatorial Optimization, 17–19 September 2012, University of Oxford, UK.
- [8] Helmar A., Chiarandini M.: *Local search heuristic for chromatic sum*, Proc. of The Metaheuristics Int Conference, MIC 2011, July 25–28, Udine, Italy.
- [9] Jin Y., Hao J.-K., Hamiez J.-P.: *A memetic algorithm for the minimum sum coloring problem*, Computers and Operations Research, Vol. 43, 2014, 318–327, DOI: 10.1016/j.cor.2013.09.019
- [10] Jin Y., Hao J.-K.: *Hybrid evolutionary search for the minimum sum coloring problem of graphs*, Information Sciences, 352–353, 2016, 15–34, DOI: 10.1016/j.ins.2016.02.051
- [11] Jin Y., Hamiez J.-P., Hao J.-K.: *Algorithms for the minimum sum coloring problem : a review*, Artificial Intelligence Review, 2016, 1–28, DOI: 10.1007/s10462-016-9485-7
- [12] Kubicka E., Schwenk A.J.: *An introduction to chromatic sums*, Proc. 17th Annual ACM Computer Science Conf., 1989, 39–45.
- [13] Li C.-M., Fang Z., Xu K.: *Combining maxsat reasoning and incremental upper bound for maximum clique problem*, Proc. 2013 IEEE 25th Int. Conf. on Tools with Artificial Intelligence, IC-TAI, 2013, 939–946.
- [14] Li Y., Lucet C., Moukrim A., Sghiouer K.: *Greedy algorithms for the minimum sum coloring problem*, Proc. Int. Workshop “Logistique et transports”, March 22–24, Sousse, Tunisia, 2009.
- [15] Lecat C., Lucet C., Li C.-M.: *New lower bound for the minimum sum coloring problem*, Proc. Thirty First Conf. on Artificial Intelligence, AAI-17, February 4–9, San Francisco, CA, USA, AAAI Press, 2017, 853–859.
- [16] Małafiejski M.: *Sumacyjne kolorowanie grafów*, [w:] Kubale M. (red.): *Optymalizacja dyskretna. Modele i metody kolorowania grafów*, WNT, Warszawa, 2002, 93–111.
- [17] Mohamed D.S., El Bernoussi S.: *Max-Min ant system for the sum coloring problem*, Proc. International Conference on Communications, Computing and Control Applications, CCCA 2011, March 3–5, Hammamet, Tunisia, 2011, 1–4.
- [18] Moukrim A., Sghiouer K., Lucet C., Li Y.: *Lower bounds for the minimal sum coloring problem*, Electronic Notes in Discrete Mathematics, Vol. 36, 2010, 663–670, DOI: 10.1016/j.endm.2010.05.084
- [19] Wang Y., Hao J.-K., Glover F., Lü Z. (2013): *Solving the minimum sum coloring problem via binary quadratic programming*. CoRR abs/1304.5876
- [20] Wu Q., Hao J.-K.: *An effective heuristic algorithm for sum coloring of graphs*, Computers and Operations Research, Vol. 39, No. 7, 2012, 1593–1600, DOI: 10.1016/j.cor.2011.09.010
- [21] Wu Q., Zhou Q., Jin J., Hao J.-K.: *Minimum sum coloring for large graphs with extraction and backward expansion search*, Applied Soft Computing, Vol. 62, 2018, 1056–1065,
DOI: 10.1016/j.asoc.2017.09.043

E.2. Odporne kolorowanie grafów [B34, B37]

Kolejnym badanym problemem kolorowania wierzchołków grafu, było kolorowanie odporne (RGCP) [7], w którym istnieje możliwość modelowania relacji sąsiedztwa wierzchołków przez liczby ułamkowe $0 < P(u,v) < 1$, przypisane krawędziom grafu (u,v) i reprezentujące prawdopodobieństwo wystąpienia sąsiedztwa. Mając dane wszystkie wartości $P(u,v)$ można zdefiniować wymagania dotyczące dopuszczalności rozwiązania: poszukiwane jest bezkonfliktowe pokolorowanie wierzchołków połączonych krawędziami E z wartością $P(u,v)=1$, oraz takie pokolorowanie wierzchołków krawędzi dopełniających \bar{E} z wartościami $0 < P(u,v) < 1$, w którym tolerowana jest pewna liczba konfliktów, pod warunkiem, że nie jest przekroczony tzw. próg sztywności (*rigidity level*). Dla ustalonej liczby kolorów, pokolorowanie z niższym progiem sztywności jest bardziej odporne (*robust*). W ogólności warunki dopuszczalności rozwiązania mogą

być wyrażone poprzez liczbę kolorów użytych, aby uzyskać dany próg odporności, lub górnego ograniczenia na funkcję kosztu odpowiadającą określonej progowi odporności, bez względu na liczbę wykorzystanych kolorów.

Podobnie jak problem GCP również RGCP jest NP-trudny [7]. W literaturze nie jest znany algorytm r -aproxymacyjny dla tego problemu. Badania prowadzone w tym obszarze obejmują algorytmy i metaheurystyki [1,2,3,5,6], nowe sformułowania problemu dla określonych zastosowań, kombinacje systemów odpornych i rozmytych. W pracy [B34] zaproponowano nowe ogólne ujęcie problemu RGCP, w którym $P(u,v)$ oznacza karę przypisaną sytuacji, gdy sąsiednie wierzchołki są w tym samym kolorze. Zdefiniowano próg odporności (*robustness threshold*) :

$$RT(c) = \frac{RRT(c)}{100\%} \sum_{(e \in \bar{E}) \wedge (p((u,v)) > 0)} p_{uv} ,$$

gdzie :

$RRT(c)$ – jest względnym progiem odporności (*relative robustness threshold*) zakładanym przez projektanta systemu i wyrażony w [%].

Stąd nasz cel optymalizacji sprowadza się do znalezienia kolorowania bezkonfliktowego dla wszystkich $P(u,v)$, spełniającego nierówność:

$$\sum_{((u,v) \in \bar{E}) \wedge (c(u) \neq c(v)) \wedge (p(u,v) > 0)} p_{uv} \geq RT(c)$$

Względna odporność równa 100% oznacza pokolorowanie bezkonfliktowe zarówno krawędzi E jak i \bar{E} , co jest zgodne z intuicją.

W badaniu problemu RGCP wykorzystano dwie metaheurystyki SA (*Simulated Annealing*) i TS (*Tabu Search*) oraz ich równoległe odpowiedniki PSA i PTS. Jako problemy testowe posłużyły grafy DIMACS o znanych liczbach chromatycznych [8,9,10], w których pewna liczba krawędzi E została losowo przeniesiona do zbioru \bar{E} . W dotychczasowych badaniach wykorzystywano grafy losowe o nieznanymi wartościach $\chi(G)$. Zdefiniowano funkcję kosztu i eksperymentalnie wyznaczono parametry PSA i PTS. Następnie wyznaczono dwa zadania dla obu metod i ich wariantów: 1) znalezienie pokolorowania o maksymalnej wartości RR dla zadanej liczby kolorów; 2) wyszukanie kolorowania odpornego z minimalną liczbą kolorów dla zadanej wartości RR. Wyniki obu eksperymentów przedstawiono na 4 wykresach słupkowych. Algorytm SA jest prostszy od TS, znacznie szybszy dla dużych grafów i jego siła zależy w większym stopniu od randomizacji niż TS. Wykazano, że w większości przypadków zadania 1) algorytm TS przeważa nad SA jakością rozwiązania, osiągając RR nie mniejsze niż 95%. Gęstsze grafy są trudniejsze do pokolorowania. W zadaniu 2) żadna metoda nie jest dominująca, ale ich równoległe wersje dają rozwiązania z nieco mniejszą liczbą kolorów. PTS działa nieco szybciej niż TS, ale PSA znacznie wolniej niż SA. Rozwiązania generowane przez PTS są bardziej powtarzalne niż PSA i SA.

W rozszerzonej wersji artykułu [B37] znajdują się dodatkowo badania RGCP przy pomocy algorytmu PEA. Zastosowano znane wcześniej operatory krzyżowania dla GCP: CEX, GPX, SPPX i po raz pierwszy BCX (*Best Coloring Crossover*) [5] oraz mutację FF. W pierwszym doświadczeniu wykazano przewagę PEA nad EA. W drugim eksperymencie dla zmodyfikowanych grafów *myciel7*, *games120* i *le450.15b* porównano czas obliczeń i liczbę operacji wymaganych dla znalezienia bezkonfliktowego rozwiązania z czterema operatorami krzyżowania. Zwycięzcą porównania ze względu na czas obliczeń okazał się operator CEX, a najmniej iteracji wymagał najbardziej złożony BCX, którego iteracja trwa najdłużej. Ze względu na jakość tego operatora proponuje się włączyć go do dalszych badań. W ostatnim teście dla zmodyfikowanych 6 grafów DIMACS zbadano wpływ rosnącego udziału krawędzi z $0 < P(u,v) < 1$ na pokolorowanie. Zaobserwowano malejący koszt kolorowania i liczbę użytych kolorów (poniżej $\chi(G)$). Dla stosunkowo rzadkich grafów liczba konfliktów rośnie, ale w przypadku grafów gęstszych (*queen5.5*, gęstość 53,3%) liczba ta maleje.

Literatura:

- [1] Archetti, C., Bianchessi N, Hertz, A.: A branch-and-price algorithm for the robust graph coloring problem. Les Cahiers du Gerad, G-2011–75, Montreal, 2011.
- [2] Lim A., Wang F.: *Meta-heuristic for robust graph coloring problem*, IEEE Int. Conference on Tools with Artificial Intelligence, ICTAI, 2004.
- [3] Lim, A., Wang, F.: *Robust graph coloring for uncertain supply chain management*, Proc. 38th Annual Hawaii Int. Conf. on System Science, HICSS 2005, IEEE, 81b (2005)
- [4] Wang, F., Xu, Z.: *Metaheuristics for robust graph coloring*, Journal of Heuristics 19(4), 2013, 529–548.
- [5] Myszkowski, P.B.: *Solving scheduling problems by evolutionary algorithms for graph coloring problem*, [in :] Xhafa F., Abraham A.: *Metaheuristics for scheduling in industrial and manufacturing applications*, Studies in Computational Intelligence, Vol. 128, 2008, 145–167.
- [6] Xu, M., Wang, Y., Wei, A.: *Robust graph coloring based on the matrix semi-tensor product with application to examination timetabling*, Control Theory and Technology 12(2), 2014, 187–197.
- [7] Yáñez J., Ramirez J.: *The robust coloring problem*, European Journal of Operational Research, Vol. 148, No. 3, 2003, 546–558.
- [8] COLOR web site. Available at: <http://mat.gsia.cmu.edu/COLOR/instances.html> .
- [9] DIMACS ftp site. Available at: <ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/> .
- [10] COLORING3 web site. Available at: <http://mat.gsia.cmu.edu/COLORING03>

E.3. Problemy permutacyjne i podziałowe [B21,B22,B28,B36,B40]

W artykule [B21] zaprezentowano wyniki badań hybrydowego algorytmu genetycznego dla problemu szeregowania zadań typu *open-shop*. Wykorzystano reprezentacje uszeregowania w postaci permutacji i permutacji z powtórzeniami. Wykorzystano dedykowane dla tego problemu operatory krzyżowania i mutacji, wspomagając je dodatkowym mechanizmem w postaci szeregowania rozwiązań uzyskanych w danym kroku algorytmu ewolucyjnego algorytmem *LPT* (*Longest Processing Time*) w dwóch wariantach: *LPT-task* i *LPT-machine*. Zaproponowana hybrydyzacja poprawiła wyniki samego EA, przy czym bardziej efektywny okazał się mechanizm szeregowania *LPT-task*. Stwierdzono jednak, że uzyskane tą drogą wyniki nie mogą konkurować z wynikami znanymi z literatury [1-2,4-6,8-12].

W pracy [B22] przeprowadzono pierwsze badania nad zastosowaniem do kolorowania grafów równoległego symulowanego wyżarzania (PSA). Większa część pracy dotyczy konstrukcji i optymalnego dostrojenia algorytmu PSA. Czas wyznaczenia przez PEA i PSA rozwiązania optymalnego dla grafów o znanych liczbach chromatycznych zależy w dużym stopniu od badanego grafu i żaden algorytm nie wykazuje zdecydowanej przewagi.

Dla probabilistycznego problemu komiwojażera PTSP [3] został skonstruowany hybrydyzowany algorytm ewolucyjny [B28]. W odróżnieniu od klasycznego problemu TSP [7] w problemie PTSP miasta na trasie komiwojażera są odwiedzane z pewnym, z góry określonym, prawdopodobieństwem, a zadanie polega na znalezieniu w grafie takiej trasy *a priori*, która minimalizuje wartość oczekiwaną kosztu ścieżki. Badania porównawcze trzech wariantów algorytmu genetycznego z hybrydyzacją w postaci wyszukiwania lokalnego wykonano na bazie biblioteki TSPLib [13].

W pracach [B36, B40] równoległe metaheurystyki: wyszukiwanie z listą tabu (TS) i symulowane wyżarzanie (SA) wraz z kilkoma wariantami były testowane na zbiorze kilku znanych problemów podziałowych, takich jak problem klasteryzacji grafu, podział grafu na kliki i kolorowanie. We wszystkich problemach przestrzeń rozwiązań składa się z podziałów zbioru wierzchołków posiadających określone właściwości. Dedykowana aplikacja zawiera dwa algorytmy sekwencyjne

i dziewięć równoległych/ hybrydowych. Z badań eksperymentalnych z udziałem grafów losowych wyprowadzono wnioski dotyczące dostrojenia metaheurystyk i jakości otrzymanych rozwiązań.

Artykuł [B40] zawiera ponadto pierwszą w literaturze charakterystykę 18 grafów z repozytorium DIMACS [14] ze względu zdefiniowaną liczbę podziałową grafu $\psi_k(G)$, $k \geq 2$, oszacowaną eksperymentalnie za pomocą metaheurystyki *H-SP (Hybrid Sequential-Parallel)*, najbardziej efektywnej ze względu na jakość rozwiązania. Szersza i dokładniejsza charakterystyka grafów DIMACS za pomocą liczby podziałowej stanowi otwarty problem badawczy.

Literatura :

- [1] Guéret C., Jussien N.: *Combining AI/OR techniques for solving Open Shop problems*, CP-AI-OR'99 (1999) 25–26
- [2] Hong T.-P., Huang P.-Y., Horng G.: *Using the LPT and Palmer approaches to solve group flexible flow-shop problems*, Int. J. Computer Science and Network Security 6 (2006) 98–104
- [3] Jaillet P.: *Probabilistic Traveling Salesman Problems*, PhD thesis, MIT, Cambridge 1985.
- [4] Khuri S., Miryala S.R.: *Genetic algorithms for solving open shop scheduling problem*, Proc. EPIA'1999, Lect. Notes in Comp. Sci. 1695 (1999), 357–368
- [5] Liaw C-F.: *A hybrid genetic algorithm for the open shop scheduling problem*, European Journal of Operational Research 124 (2000) 28–42
- [6] Louis S.J., Xu Z.: *Genetic algorithms for open shop scheduling and re-scheduling*, 11th ISCA'96 (1996) 99–102
- [7] Michalewicz Z.: *Evolutionary Algorithms + Data Structures = Evolutionary Programs*, 3rd ed., Springer, 1996.
- [8] Oliver M., Smith D. J., Holland J. R. C. (1987) *A study of permutation crossover operators on the travelling salesman problem*, Proc. 2nd Int. Conf. on Genetic Algorithms and Their Application, 224–230
- [9] Prins C.: *Competitive genetic algorithm for the open-shop scheduling problem*, Mathematical Methods of Operations Research 52 (2000), 389–411.
- [10] Puente J., Diez H.R., Varela R., Vela C.R., Hidalgo L.P.: *Heuristic rules and genetic algorithms for open shop scheduling problem*, Proc. CAEPIA'2003, Lect. Notes in Comp. Sci. 3040 (2004), 394–403.
- [11] Syswerda G.: *Scheduling optimization using genetic algorithms*. [in:] Davis L. (ed.): Handbook of Genetic Algorithms. Van Nostrand Reinhold (1991), 332–349.
- [12] Taillard E.: *Benchmarks for basic scheduling problems*, European Journal of Operational Research 64 (1993), 278–285
- [13] TSPLib, <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/>
- [14] DIMACS ftp site. Available at: <ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/> .

F. Projektowanie układów cyfrowych

F.1. Synteza i dekompozycja układów cyfrowych [B27, B38]

Wieloletnie doświadczenie badawcze i dydaktyczne habilitanta w obszarze syntezy układów cyfrowych oraz układów programowalnych FPGA przyniosło wynik w postaci patentów [AP1,AP2] oraz programu *PKmin* [10] wspierającego projektowanie układów Full Custom, Semi Custom i FPGA. Program *PKmin* implementuje wiele opracowanych przez prof. Adama Kaprańskiego algorytmów syntezy i dekompozycji układów cyfrowych [3-8]. Program został wykonany w ramach pracy dyplomowej przez mgr inż. Tomasza Michalskiego.

W zakresie syntezy układów cyfrowych program *PKmin* umożliwia dowolne zestawienie narzędzi składowych: algorytmów wyznaczania kombinacyjnej zależności macierzy X z wektorem y , algorytmów redukcji liczby argumentów w macierzy X , algorytmów wyznaczania bazy macierzy Y (problem NP–zupełny [5]) w przypadku syntezy układów dwumetapoziomowych itp. [1, 3-8].

W zakresie dekompozycji funkcjonalnej układów wielowejściowych wzorowano się na pakiecie *DEMAIN* opracowanym w Politechnice Warszawskiej [11]. Do algorytmu dekompozycji wprowadzono jednak pewne modyfikacje (priorytet dekompozycji równoległej w stosunku do

dekompozycji szeregowej) wspierające kompromis pomiędzy liczbą bloków LUT układu FPGA a liczbą poziomów układu. Może to skutkować nieco większą liczbą LUT przy mniejszej liczbie poziomów. Zapewniono wizualizację procesu projektowania. Zaimplementowane narzędzia syntezy i dekompozycji układów cyfrowych zostały porównane z innymi narzędziami akademickimi (*Espresso*, *DEMAIN*) i komercyjnymi: *MAX+PLUS2* (Altera), *Web Pack ISE* (Xilinx). Program *Espresso* najlepiej optymalizował projektowany układ, ale za cenę bardzo długiego czasu syntezy. W przypadkach, gdy brany jest pod uwagę czas uzyskania rozwiązania, korzystne może się okazać wykorzystanie programu *PKmin*. Dotyczy to przede wszystkim projektowania układów o małej i średniej złożoności oraz etapu prototypowania. Test dekompozycji funkcjonalnej układu polegający na projekcie konwertera kodu *BIN2BCD* (benchmark) w technologii FPGA wykazał zbliżone wyniki dekompozycji przy wykorzystaniu trybu interaktywnego *DEMAIN* i *PKmin* oraz nieco gorsze wyniki *PKmin* uzyskiwane w trybie w pełni automatycznym. Potwierdziła się opinia, że wyniki uzyskiwane przy pomocy pakietów akademickich [9-11] generują rozwiązania zdecydowanie lepsze od popularnych narzędzi komercyjnych. Opis i badania pakietu *PKmin* [10] zostały opublikowane w dwóch artykułach [B27, B38], przedstawione na konferencji [B35] oraz na seminarium w Politechnice Warszawskiej (2015).

Posiadający duże walory dydaktyczne program *PKmin* jest ogólnodostępny na stronie projektu [10].

Literatura:

- [1] De Micheli G. : *Synthesis and optimization of digital circuits*, McGrawHill, 1994.
- [2] Borowik G., Łuba T. : *Fast algorithm for attribute reduction based on the complementation of Boolean function*, [in:] *Advanced Methods and Applications in Computational Intelligence*, 25–45, Springer International, 2014.
- [3] Kaprański A.: *Wstęp do teorii przestrzeni boolowskich i ich zastosowanie w syntezie układów przełączających*, Zeszyty Naukowe Politechniki Krakowskiej, Seria: Transport, Z. 2, Kraków, 1979.
- [4] Kaprański A.: *Wstęp do teorii układów przełączających i teorii informacji*, Wydawnictwo Politechniki Krakowskiej, Kraków 1985.
- [5] Kaprański A., Skarbek W. : *Problem of searching minimum base in Boolean tables*, *Podstawy sterowania*, 257–265, 1986.
- [6] Kozakow W. D. : *Minimization of multi-variable logic functions*, *Avtomatika i Telemekhanika*, Vol. 23, No. 9, 1962. (in Russian)
- [7] Łuba T.: *Synteza układów logicznych*, Oficyna Wydawnicza PW, Warszawa 2005.
- [8] Nowicka M., Łuba T., Rawski M.: *FPGA-based decomposition of Boolean functions. Algorithms and implementation*, *Proc. 6th Int. Conference on Advanced Computer Systems*, Szczecin 1999, 502–509.
- [9] Salauyou V., Klimowicz A. : *Synteza logiczna układów cyfrowych w strukturach programowalnych*, Oficyna Wydawnicza Politechniki Białostockiej, Białystok, 2010.
- [10] PKmin WWW site : <http://www.pkmin.cba.pl> (do dn. 17.04.2018 <http://www.pkmin.za.pl>)
- [11] DEMAIN WWW site : <http://www.zpt.tele.pw.edu.pl/oprogramowanie/demain.html>

G. Podsumowanie

Na osiągnięcia naukowe poza przedstawionym cyklem publikacji mieszające się w obszarach wskazanych w podrozdziałach **5.D** – **5.F** składają się oryginalne wyniki, cytowane w światowej literaturze naukowej:

- **projekt i implementacja FPGA procesora asocjacyjnego z wieloma komparandami, który może posłużyć jako akcelerator w równoległym przetwarzaniu wybranych problemów grafowych modelowanych przez macierze sąsiedztwa [B8,B12,B25]**
- **architektura i algorytm programowania kwadratowej komórkowej sieci połączeniowej do permutowania danych w modelu SIMD [B9]**

- konstrukcja i zastosowanie równoległych i hybrydowych metaheurystyk do przetwarzania problemów kolorowania grafów (sumacyjnego, odpornego) oraz innych problemów podziałowych i permutacyjnych [B20,B21,B22,B28,B40]
- teoretyczne i eksperymentalne oszacowania liczb chromatycznych $\Sigma(G)$, $s(G)$ i liczby podziałowej $\psi_k(G)$ grafów z repozytorium DIMACS za pomocą metaheurystyk równoległych i hybrydowych [B20,B40]
- opracowanie koncepcji i implementacja programu *PKmin* do wspomaganej komputerowo syntezy i dekompozycji funkcjonalnej kombinacyjnych układów cyfrowych dla różnych technologii wykonania układów w oparciu o oryginalny w stosunku do innych dostępnych narzędzi akademickich zestaw algorytmów [B27,B38]

Najczęściej cytowane są publikacje [B8] i [B20]. Prace nad zastosowaniami procesora multiasocjacyjnego i metaheurystykami równoległymi są kontynuowane. Program *PKmin* jest stosowany w Politechnice Krakowskiej i Politechnice Warszawskiej jako narzędzie dydaktyczne.

Bibliografia prac własnych (porządek chronologiczny)

A. Wykaz opublikowanych prac naukowych i uzyskanych patentów przed uzyskaniem stopnia naukowego doktora*

- [A1] Kokosiński Z.: *O efektywności algorytmów minimalizacji funkcji logicznych*, Prace 3. Ogólnopolskiej Konferencji SEMTRAK'86, Kraków-Janowice, 108-113 (1986)
- [A2] Kokosiński Z.: *Analysis of the Sarje method for minimalization of multiple-output switching circuits*, Bulletins for Applied Mathematics, 553/87 [XLIX], 299-306 (1987) ISSN 0133-3526 (Zentralblatt MATH)
- [A3] Kokosiński Z.: *A new framework for developing modular redundancy schemes*, Proc. 3rd Int. Conference RELCOMEX'87, Książ Castle, Poland, Vol.2, 205-211 (1987)
- [A4] Kokosiński Z.: *A survey of recent results in the design of interconnection networks for multiprocessor systems*, Prace Konferencji MIKROKOMPUTER'88, Bierutowice, Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, nr 79, Seria: Konferencje, nr 36, 95-99 (1988) ISSN: 0324-9794
- [A5] Kokosiński Z.: *CELLIB. Towards editing cellular schemes with draft from OrCAD/SdT package*, Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, Seria: Konferencje, nr 89, Seria: Konferencje, nr 39, 169-171 (1991) ISSN: 0324-9794
- [A6] Kokosiński Z.: *Iterative decomposition of permutations*, Proceedings of the Second International Mathematical Miniconference, Part II (Budapest, 1988), [in:] Periodica Polytechnica, Series: Transportation Engrg., Vol. 19, Nos.1-2, 49-55 (1991) HUISSN: 0303-7800 (AMS MathSciNet MR1169111)
- [A7] Kokosiński Z.: *On generation of permutations through decomposition of symmetric groups into cosets*, BIT, Vol.30, 583-591 (1990) ISSN: 0006-3835, DOI: 10.1007/BF01933207
- [A8] Kokosiński Z.: *Układy generatorów obiektów kombinatorycznych dla systemów sekwencyjnych i równoległych*, Monografia 160, Wydawnictwo Politechniki Krakowskiej, 106 s. (1993)
- [A9] Kokosiński Z.: *Experimental performance of parallel computations in SIMD model - a case study*, Proc. 13th IASTED Int. Conference on Modeling, Identification and Control MIC'94, Grindelwald, Switzerland, 442-445 (1994) ISBN: 0-88986-183-8
- [A10] Kokosiński Z.: *Mask and pattern generation for associative supercomputing*, Proc. 12th IASTED Int. Conference on Applied Informatics AI'94, Annecy, France, 324-326 (1994) Acta Press ISBN: 0-88986-190-0
- [AP1] Kaprański A., Kokosiński Z., Mól W.: *Elektroniczny układ cyfrowy do wyboru liczby ekstremalnej zawartej w pamięci o dostępie swobodnym komputera*, Urząd Patentowy RP, Patent No.146021, 1989
- [AP2] Kokosiński Z.: *Elektroniczny układ cyfrowy do generowania permutacji zbioru danych zapisanego w pamięci RAM*, Urząd Patentowy RP, Patent No.161547, 1994

*Publikacje [A8-A10] i patent [AP2] z lat 1993-1994 są związane z tematyką rozprawy doktorskiej i dlatego zostały uwzględnione w części A wykazu chronologicznego.

B. Wykaz opublikowanych prac naukowych po uzyskaniu stopnia naukowego doktora

- [B1] Kokosiński Z.: *Algorithms for unranking combinations and their applications*, Proc. 7th IASTED/ISMM Int. Conference on Parallel and Distributed Computing and Systems PDCS'95, Washington D.C., USA, 216-224 (1995) Acta Press ISBN: 0-88986-228-1
- [B2] Kokosiński Z.: *Unranking combinations in parallel*, Proc. 2nd Int. Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'96, Sunnyvale, CA, USA, Vol. I, 79-82 (1996) CSREA Press ISBN: 0-9648666-1-7
- [B3] Kokosiński Z.: *Generation of integer compositions on a linear array of processors*, Proc. 2nd Int. Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'96, Sunnyvale, CA, USA, Vol. I, 56-64 (1996) CSREA Press, ISBN: 0-9648666-1-7
- [B4] Kokosiński Z.: *On parallel generation of combinations in associative processor architectures*, Proc. IASTED Int. Conference Parallel and Distributed Systems Euro-PDS'97, Barcelona, Spain, 283-289 (1997) Acta Press ISBN: 0-88986-225-7
- [B5] Kokosiński Z.: *An associative processor for multi-comparand parallel searching and its selected applications*, Proc. 3rd Int. Conference Parallel and Distributed Processing Techniques and Applications PDPTA'97, Las Vegas, USA, Vol. III, 1434-1442 (1997) CSREA, Press ISBN: 0-9648666-7-6
- [B6] Kokosiński Z.: *On parallel generation of set partitions in associative processor architectures*, Proc. 5th Int. Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'99, Las Vegas, USA, Vol. III, 1257-1262 (1999) CSREA, Press ISBN: 1-892512-11-4
- [B7] Kokosiński Z.: *A chromosome representation of permutations for genetic algorithms*, Proc. Int. Conference on Artificial Intelligence IC-AI'99, Las Vegas, USA, Vol. I, 65-69 (1999) CSREA Press ISBN: 1-892512-16-5
- [B8] Kokosiński Z.: *On parallel generation of t-ary trees in an associative model*, Postproc. Parallel Processing and Applied Mathematics PPAM'2001, Lecture Notes in Computer Science 2328, 228-235 (2002) Springer-Verlag, DOI: 10.1007/3-540-48086-2_25
- [B9] Kokosiński Z., Sikora W.: *An FPGA implementation of multi-comparand multi-search associative processor*, Proc. FPL'2002 (Field Programmable Logic and Applications) LNCS 2438, 826-835 (2002) Springer-Verlag, DOI: 10.1007/3-540-46117-5_85,
- [B10] Kokosiński Z.: *Square interconnection network for data permutation*, Proc. 3rd Int. Conference on Parallel Computing in Electrical Engineering PARELEC'2002, Warsaw, Poland, IEEE Computer Society Press, 44-46 (2002), DOI: 10.1109/PCEE.2002.1115195
- [B11] Kokosiński Z.: *On generation of permutations through suffix/prefix reversing in a cellular network*, Postproc. Parallel Processing and Applied Mathematics PPAM'2003, Lecture Notes in Computer Science 3019 (2004), 249-254, Springer-Verlag, DOI: 10.1007/978-3-540-24669-5_32
- [B12] Nepomniaschaya A., Kokosiński Z.: *Associative graph processor and its properties*, Proc. 4th Int. Conference on Parallel Computing in Electrical Engineering PARELEC'2004, Dresden, Germany, IEEE Computer Society Press, 297-302 (2004), DOI: 10.1109/PCEE.2004.15
- [B13] Kokosiński Z.: *A parallel dynamic programming algorithm for unranking t-ary trees*, Postproc. Parallel Processing and Applied Mathematics PPAM'2003, Lecture Notes in Computer Science 3019 (2004), 255-260, Springer-Verlag, DOI: 10.1007/978-3-540-24669-5_33

- [B14] Kwarciany K., Kokosiński Z.: *Równoległe algorytmy genetyczne w zastosowaniu do problemu kolorowania wierzchołków grafu*, Prace IT'2004 (Technologie Informacyjne), Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej, Seria: Technologie Informacyjne, No.5, 747-754 (2004) ISBN: 83-917681-5-5
- [B15] Kokosiński Z., Kołodziej M., Kwarciany K.: *Parallel genetic algorithm for graph coloring problem*, Proc. ICCS'2004 (Computational Science), Lecture Notes in Computer Science 3036, 217-224 (2004) Springer-Verlag, DOI: 10.1007/978-3-540-24685-5_27
- [B16] Bubniak G., Góralczyk M., Karp M., Kokosiński Z.: *A hardware implementation of a generator of (n,k) -combinations*, Proc. IFAC Workshop Programmable Digital Systems, PDS'2004, Kraków, Poland, 228-231 (2004) ISBN 83-908409-8-7
- [B17] Kokosiński Z., Kwarciany K., Kołodziej M.: *Efficient graph coloring with parallel genetic algorithms*, Computing and Informatics 24, 123-147 (2005), ISSN: 1335-9150
- [B18] Kokosiński Z.: *Effects of versatile crossover and mutation operators on evolutionary search in partition and permutation problems*, Proc. IIS: IIPWM'2005, Gdańsk, Poland, Advances in Soft Computing, 299-308 (2005), Springer-Verlag, DOI: 10.1007/3-540-32392-9_31
- [B19] Kokosiński Z.: *A new algorithm for generation of exactly m -block partitions in associative model*, Postproc. Parallel Processing and Applied Mathematics PPAM'2005, Lecture Notes in Computer Science 3911, 67-74 (2006) Springer-Verlag, DOI: 10.1007/11752578_9
- [B20] Kokosinski Z., Kwarciany K.: *On sum coloring of graphs with parallel genetic algorithms*, Proc. ICANNGA'2007 (Adaptive and Natural Computing Algorithms), Lecture Notes in Computer Science 4431, 211-219 (2007), Springer-Verlag, DOI: 10.1007/978-3-540-71618-1_24
- [B21] Kokosiński Z., Studzienny Ł.: *Hybrid genetic algorithms for the open-shop scheduling problem*, Int. Journal of Computer Science and Network Security, Vol.7, No.10, 136-145 (2007) ISSN: 1738-7906
- [B22] Łukasik S., Kokosinski Z., Świętoń G.: *Parallel simulated annealing algorithm for graph coloring problem*, Postproc. Parallel Processing and Applied Mathematics PPAM'2007, Lecture Notes in Computer Science 4967, 229-238 (2008), Springer-Verlag, DOI: 10.1007/978-3-540-68111-3_25
- [B23] Kokosiński Z.: *On generation of derangements, partial derangements and permutations*, Postproc. Parallel Processing and Applied Mathematics PPAM'2007, Lecture Notes in Computer Science 4967, Springer-Verlag ISSN: 0302-9743, 219-228 (2008), DOI: 10.1007/978-3-540-68111-3_24
- [B24] Kokosiński Z., Halesiak P.: *FPGA generators of combinatorial configurations in a linear array model*, Proc. 7th Int. Symposium on Parallel and Distributed Computing ISPDC'2008, Kraków, Poland, IEEE Computer Society Press, 223-227 (2008) DOI: 10.1109/ISPDC.2008.48
- [B25] Kokosiński Z., Malus B.: *FPGA implementations of a parallel associative processor with multi-comparand multi-search operations*, Proc. 7th Int. Symposium on Parallel and Distributed Computing ISPDC'2008, Kraków, Poland, IEEE Computer Society Press, 444-448 (2008), DOI: 10.1109/ISPDC.2008.42
- [B26] Kokosiński Z.: *Parallel enumeration of t -ary trees in ASC SIMD model*, Int. Journal of Computer Science and Network Security, Vol. 11, No. 12, 38-49 (2011) ISSN : 1738-7906

- [B27] Kokosiński Z., Michalski T.: *Synteza dwupoziomowych układów kombinacyjnych w programie PKmin*, Czasopismo Techniczne, Seria: Automatyka, Politechnika Krakowska, 1-AC/2012, 93-103, zeszyt 25, rok 109 (2012) ISSN: 0011-4561, DOI: 10.4467/2353737XCT.14.007.1784
- [B28] Kielkowicz K., Kokosiński Z.: *Algorytm hybrydowy dla probabilistycznego problemu komiwojażera*, Czasopismo Techniczne, Seria: Automatyka, Politechnika Krakowska, 1-AC/2012, 115-126, zeszyt 25, rok 109 (2012) ISSN: 0011-4561, DOI: 10.4467/2353737XCT.14.008.1785
- [B29] Kokosiński Z.: *Parallel metaheuristics in graph coloring*, Visnik Nacionalnogo Universitetu „Lvivska Politechnika” - Kompiuterni nauki ta informacijni technologii, No. 744, 209-214 (2012) ISSN: 0321-0499
- [B30] Kokosiński Z.: *A parallel dynamic programming algorithms for unranking set partitions*, Technical Transactions, Seria: Automatic Control, Cracow University of Technology, 4-AC/2013, 29-38, issue 28, year 110 (2013) ISSN: 0011-4561, DOI: 10.4467/2353737XCT.14.055.3963
- [B31] Kokosiński Z., Wójcik S.: *A comparison of SW/HW implementations of stream cipher encoders*, Technical Transactions, Seria: Automatic Control, Cracow University of Technology, 4-AC/2013, 83-92, issue 28, year 110 (2013), ISSN: 0011-4561, DOI: 10.4467/2353737XCT.14.059.3967
- [B32] Kokosiński Z., Domagała P.: *Solving graph coloring problem with parallel evolutionary algorithms in a mesh model*, Proc. Federated Conference on Computer Science and Information Systems FedCSIS'2014 (Position papers), Annals of Computer Science and Information Systems, PTI, Vol.3, 21-28, (2014), DOI: 10.15439/2014F391
- [B33] Chakraborty G., Horie S., Yokoha H., Kokosiński Z.: *Minimizing sensors for system monitoring - a case study with EEG signals*, 2nd IEEE International Conference on Cybernetics CYBCONF'2015, Gdynia, Poland, 24-26 June 2015, IEEE Xplore, 206–211 (2015), DOI: 10.1109/CYBConf.2015.7175933
- [B34] Kokosiński Z., Ochał Ł.: *Evaluation of metaheuristics for robust graph coloring problem*, Proc. Federated Conference on Computer Science and Information Systems FedCSIS'2014, Łódź, Poland, 13-16 September, IEEE Xplore, Annals of Computer Science and Information Systems, PTI, Vol.5, 519-524 (2015) DOI: 10.15439/2015F293
- [B35] Kokosiński Z., Michalski T.: *Synthesis and functional decomposition of digital circuits with PKMIN*, Proc. 3rd Int. Conference on Automatic Control and Information Technologies, ICACIT'2015, Kyiv, Ukraine, December 11-13, KPI Press, 20-23 (2015)
- [B36] Kokosiński Z., Bała M.: *Testing of parallel metaheuristics for graph partitioning problems*, Proc. Federated Conference on Computer Science and Information Systems FedCSIS'2016 (Position papers), Annals of Computer Science and Information Systems, PTI, Vol. 9, 79-85, (2016) DOI: 10.15439/2016F414
- [B37] Kokosiński Z., Ochał Ł., Chrzęszcz G.: *Parallel metaheuristics for robust graph coloring problem*, [in:] Fidanova S. (ed): *Recent Advances in Computational Optimization*, Studies in Computational Intelligence, Vol. 655, 285-302, Springer International Publishing (2016), DOI: 10.1007/978-3-319-40132-4_16
- [B38] Michalski T., Kokosiński Z.: *Functional decomposition of combinational logic circuits with PKmin*, Technical Transactions, Cracow University of Technology, 2-E/2016, 191-202, (2016) ISSN: 0011-4561, DOI: 10.4467/2353737XCT.16.257.6056
- [B39] Kokosiński Z.: *On generation of permutations of m out of n elements*, Information Processing Letters, Vol. 124, 1-5 (2017), DOI: 10.1016/j.ipl.2017.04.001

[B40] [Kokosiński Z.](#), Pijanowski M.: *Comparison of Approximate Methods for Partitioning of Digital Circuits*, Proc. 4th Int. Conference on Automatic Control and Information Technologies, ICACIT'2017, Kraków, Poland, December 14-16, Cracow University of Technology, 102-108, CD-ROM (2017)

[B41] [Kokosiński Z.](#), Bała M.: Solving graph partitioning problems with parallel metaheuristics, [in:] Fidanova S. (ed): *Recent Advances in Computational Optimization, Studies in Computational Intelligence*, Vol. 717, 89-105, Springer International Publishing (2018), DOI: 10.1007/978-3-319-59861-1_6 (dostęp online od dnia 28.06.2017 r.)

Zbiornicze zestawienie osiągnięć

Osiągnięcia kandydata zgodnie z wymogami Rozporządzenia Ministra Nauki i Szkolnictwa Wyższego z dnia 1 września 2011 roku w sprawie kryteriów oceny osiągnięć osoby ubiegającej się o nadanie stopnia doktora habilitowanego (Dz.U. Nr 196, poz. 1165), zestawiono w tablicy V.

Tablica V Kryteria osiągnięć osoby ubiegającej się o nadanie stopnia doktora habilitowanego (na podstawie Załączników 2, 3, 7 i 8 wniosku)

Kryterium według § 3 p.4, §4 i §5 Rozporządzenia	Spełnienie kryterium/liczba
Publikacje naukowe w czasopismach znajdujące się w bazie Journal Citation Reports (JCR)	9 (1 – przed doktoratem)
Zrealizowane oryginalne osiągnięcie projektowe, konstrukcyjne i technologiczne	3
Udzielone patenty (krajowe)	2 (1 – przed doktoratem)
Monografie	1
Publikacje naukowe w czasopismach międzynarodowych lub krajowych	19 (3 – przed doktoratem)
Opracowania zbiorowe, katalogi zbiorów, dokumentacja prac badawczych, ekspertyz, utworów i dzieł artystycznych	tak
Sumaryczny impact factor według listy Journal Citation Reports (JCR), zgodnie z rokiem opublikowania	4,220
Liczba cytowań publikacji według bazy Web of Science (WoS)	68
Indeks Hirscha według bazy Web of Science (WoS)	5
Kierowanie międzynarodowymi i krajowymi projektami badawczymi oraz udział w takich projektach	3 – kierowanie (krajowe) 3 – udział (krajowe)
Międzynarodowe i krajowe nagrody za działalność naukową lub artystyczną	3 (krajowe)
Wygłoszenie referatów na międzynarodowych i krajowych konferencjach / Aktywny udział w międzynarodowych i krajowych konferencjach naukowych	32 (4 – przed doktoratem) / 23 (komitety naukowe konferencji międzynarodowych)
Uczestnictwo w programach europejskich oraz innych programach międzynarodowych i krajowych	2 (PO Kapitał Ludzki – UE, CEEPUS – NAWA)
Udział w komitetach organizacyjnych międzynarodowych i krajowych konferencji naukowych	2 (międzynarodowe)
Otrzymane nagrody i wyróżnienia	6 (krajowe)
Kierowanie projektami realizowanymi we współpracy z naukowcami z innych ośrodków polskich i zagranicznych oraz we współpracy z przedsiębiorcami	nie
Udział w komitetach redakcyjnych i radach naukowych czasopism	1
Członkostwo w międzynarodowych i krajowych organizacjach oraz towarzystwach naukowych	4
Osiągnięcia dydaktyczne i w zakresie popularyzacji nauki lub sztuki	tak
Opieka naukowa nad studentami i lekarzami w toku specjalizacji	55 – dyplomanci
Opieka naukowa nad doktorantami w charakterze opiekuna naukowego lub promotora pomocniczego	nie
Staże w zagranicznych i krajowych ośr. naukowych lub akademickich	36 m-cy (uniwersytet zagr.)
Wykonanie ekspertyzy lub innego opracowania na zamówienie	11 - (CEEPUS – NAWA)
Recenzowanie publikacji w czasopismach krajowych/zagranicznych	2/18
Recenzowanie publikacji w czasopismach międzynarodowych z listy Journal Citation Reports (JCR)	14