Acta Universitatis Sapientiae

Informatica

Volume 9, Number 1, 2017

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 5–16

DOI: 10.1515/ausi-2017-0001

Antal Iványi (1942–2017)

Zoltán KÁSA
Sapientia Hungarian University of Transylvania
email: kasa@ms.sapientia.ro

István FEKETE
Loránd Eötvös University
email: fekete@inf.elte.hu

Professor Antal Iványi, an active and always helpful member of our editorial board, passed away on January 8, 2017 at the age of 75. He was born in Kecskemét on January 23, 1942. After graduating from the Faculty of Chemical Engineering Veszprém in 1965, he obtained a degree in Mathematics from Eötvös Loránd University (ELTE) Budapest in 1969. From 1971 he taught

Computing Classification System 1998: A.0.
Mathematics Subject Classification 2010: 01A70
Key words and phrases: Antal Iványi

at the Department of Numerical and Computational Mathematics (since 2003 Faculty of Informatics) of ELTE Budapest until his retiring in 2012.

His interest turned to computers and their applications early. Initially, he taught mathematics students subjects such as Computational Mathematics, Programming of Digital Electronic Computers, ALGOL Programming with Chemical Applications etc. Between 1972 and 1975, and later in 1983-84, he studied and worked at the Moscow State University. He received his university doctor degree in 1972, afterwards he earned his CSc (candidate of science) in 1978, and his DSc (doctor of science) in 1984, in Mathematics and Computer Science.

He initiated and organized between 1984 and 1989 the international conference of young program designer at the ELTE Budapest.

He was not only a professor and researcher, but an outstanding educator, educational organizer, book writer, translator and editor. His books and edited volumes contribute significantly to the development of Hungarian computer science education. We mention here only the most important translations and volumes coordinated by Professor Iványi:

- Cormen–Leiserson–Rivest: Introduction to Algorithms,
- Cormen–Leiserson–Rivest–Stein: Introduction to Algorithms,
- Donald E. Knuth: The Art of Computer Programming (issues of vol. 4),
- Algorithms of Informatics (three volumes in Hungarian and English).

Professor Iványi was also a former competitor of the University Athletics Club of Budapest (BEAC). He liked chess, bridge and sudoku too.

For his outstanding work he received the Neumann Prize awarded by the John von Neumann Computer Society (Budapest) in 2005.

Despite his illness, he remained active until the end of his life. His last edited book has been left unfinished.

With a terrible feeling of pain and loss, we say goodbye to our colleague and friend. We shall treasure his memory!

On behalf of the Editorial Board
Z. Kása

In memoriam

When I try to invoke the memory of our colleague and friend, Anti Iványi (Toncsi, Tony), I can recall four situations of our long-life relationship.

Anti, the teacher

It goes back to 1971 when I first met him. I was a second-year student when Anti (not yet 30) gave us a course in Numerical mathematics. It was a reve-

lation for us at that time that commonly known constants can be computed to many digits or transcendental equations can be solved approximately by root-finding algorithms. At the end of the semester the enthusiastic teacher could give a mark "five" (excellent) to his dedicated student. This mark and his signature is a curiously valuable entry in my university record book.

Anti, the editor of the book *Algoritmusok*

A great adventure of his active life was to manage the Hungarian translation of the book *Introduction to Algorithms* by Cormen et al. This book brought the wonderful experience of the common creative work with Anti. We had been co-designers of the cover of the book. Remember that A. Calder's "Big Red" appearing on the original book cover lets the reader associate to the data structure "tree". During a long conversation an idea popped in our heads: for the cover of the Hungarian edition a "Hungarian tree" would be most suitable. The next thought already was Csontváry's "The Lonely Cedar". However, to translate this idea into action turned out to be extremely hard (copyright problems, poor quality diapositives) and any person other than Anti presumably would gave up the realization. Finally, in 1997 the *Algoritmusok* appeared with its attractive cover.

Anti, the tennis partner

From the second part of the '80-s we were bound by the love of tennis. Through nearly 30 years we played a lot against each other and together in a double. Anti as a double partner was really amazing. From the very first moment until the last one it was unquestionably a team on the court. The responsibility and the assistance for each other was achieved beside him on a high level. He never exhausted, which was quite unbelievable to see, not even in an extremely hot weather. He always wanted to win and he never gave up.

Anti, the trusted friend

His long struggle with illness began in 2004 when Anti had the first surgery. He spent a longer period in the hospital and I visited him two or three times in a week. He asked me to manage his extensive electronic correspondence. I was allowed to answer the easy letters independently (with naming the situation). However, I had to print out the more complicated ones, and during the visiting time Anti compiled the similarly complex answers using a red pen. In the third week, I was heavily interested in his healing, as I felt. Coming back to

the university he mentioned in laughs several times that my letters certainly improved his human relations. After a long and desperate battle with illness in January this year, Anti passed away. He was a unique man who always tried to be fairly similar to majority but he remained an individual, from the distance he kept, a lovable individual.

István Fekete

Antal Iványi's publications

Scientific papers

1. A. Iványi, N. Fogarasi, On partial sorting in restricted rounds, *Acta Univ. Sapientiae, Informatica*, **9,** 1 (2017) 17–34.

2. A. Iványi, S. Pirzada, F. A. Dar, Tripartite graphs with given degree set *Acta Univ. Sapientiae, Informatica*, **7,** 1 (2015) 72–106.

3. A. M. Iványi, Z. Kása. Parallel partial ranking, *Applied Discrete Mathematics and Heuristic Algorithms* (Russia), **1,** 3 (2015) 57–76.

4. Iványi Antal, Kása Zoltán, Foksorozatok párhuzamos leszámlálása, *Alkalmazott Matematikai Lapok*, **31** (2014) 41–98.

5. A. Iványi, Reconstruction of score sets *Acta Univ. Sapientiae, Informatica* **6,** 2 (2014) 210–229.

6. B. A. Chat, S. Pirzada, A. Iványi, Recognition of split-graphic sequences, *Acta Univ. Sapientiae, Informatica*, **6,** 2 (2014) 252–286.

7. T. A. Chishti, G. Zhou, S. Pirzada, A. Iványi, On vertex independence number of uniform hypergraphs, *Acta Univ. Sapientiae, Informatica*, **6,** 1 (2014) 132–158.

8. H. A. Ganie, S. Pirzada, A. Iványi, Energy, Laplacian energy of double graphs and new families of equienergetic graphs, *Acta Univ. Sapientiae, Informatica*, **6,** 1 (2014) 89–116.

9. A. Iványi, J. Elek, Degree sets of tournaments, *Studia Univ. Babeş-Bolyai, Inform.* **59,** *Special Issue* 1, (2014), 150—164.

10. A. Iványi, Leader election in synchronous networks, *Acta Univ. Sapientiae, Math.* **5,** 1 (2013), 54—82.

11. A. Iványi, L. Lucz, G. Gombos, T. Matuszka, Parallel enumeration of degree sequences of simple graphs. II *Acta Univ. Sapientiae, Informatica* **5,** 2 (2013) 245-270.

12. A. Iványi, L. Lucz, T. Matuszka, G. Gombos, Score sets in multitournaments I. Mathematical results, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **40** (2013) 307–320.

13. A. Iványi, Z. Kása, Prism complexity of matrices, *Annales Univ. Sci. Budapest., Sect. Comp.* **39** (2013) 181–202.

14. A. Iványi, L. Lucz, T. Matuszka, S. Pirzada, Parallel enumeration of degree sequences of simple graphs, *Acta Univ. Sapientiae Informatica* **4,** 2 (2012) 260–288.

15. A. Iványi, J. E. Schoenfield, Deciding football sequences, *Acta Univ. Sapientiae Informatica*, **4,** 1 (2012) 130–183.

16. A. Iványi, Degree sequences of multigraphs, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **37** (2012) 195–214.

17. A. Iványi, S. Pirzada, A. Shah Nasir, Imbalances of bipartite multitournaments, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **37** (2012) 215–228.

18. S. Pirzada, A. Iványi, Minimal digraphs with given imbalance sequence, *Acta Unuversitatis Sapientiae Mathematica* **4,** 1 (2012) 86–101.

19. A. Iványi, L. Lucz, Multigráfok foksorozatai (Degree sequences of multigraphs), *Alkalmazott Mat. Lapok* **29,** (2012) 1–52 .

20. A. Iványi, I. Kátai, Testing of random matrices, *Acta Univ. Sapientiae Informatica*, **3,** 1 (2011) 99–126.

21. Iványi Antal, Németh Zsolt, List coloring of Latin and Sudoku graphs, In: Horia F Pop, Antal Bege (Ed.) *8th Joint Conference on Mathematics and Computer Science, MaCS 2010, Komárno, Slovakia, July 14–17, 2010, selected papers.* 416 p. Komárno, Slovakia, 2010 July 14–17. Győr, Novadat, 2011. pp. 23–34. (ISBN: 978-963-9056-38-1)

22. A. Iványi, Reconstruction of complete interval tournaments. II, *Acta Univ. Sapientiae Mathematica* **2,** 1 (2010) 47–71.

23. S. Pirzada, T. A. Naikoo, U. Samee, A. Iványi, Imbalances in directed multigraphs, *Acta Univ. Sapientiae Mathematica* **2,** 2 (2010) 137–145.

24. P. Fornai, A. Iványi, FIFO anomaly is unbounded *Acta Univ. Sapientiae Informatica*, **2,** 1 (2010) 80–89.

25. A. Iványi, Density of safe matrices, *Acta Univ. Sapientiae Mathematica* **1,** 2 (2009) 121–142.

26. A. Iványi, Reconstruction of complete interval tournaments, *Acta Univ. Sapientiae Informatica*, **1,** 1 (2009) 71–88.

27. M. Horváth, A. Iványi, Growing perfect cubes, *Discrete Mathematics* **308,** 19 (2008) 4378–4388.

28. M-C. Anisiu, A. Iványi, Two-dimensional arrays with maximal complexity, *Pure Mathematics and Applications* **17,** 3-4 (2006) 197–204.

29. A. Iványi, Maximal tournaments, *Pure Mathematics and Applications* **13,** 1-2 (2002) 171–183.

30. A. M. Iványi, Construction of three-dimensional perfect matrices, *Ars Combinatoria* **29(C)** (1990) 33–40.

31. A. Iványi A, L. Kauitar, Simultaneous solution of the traveling salesman problem and the packing problem, *Vestnik Moskovskogo Universiteta Seryia 15 Vycislitelnaya Matematika i Kibernetika* **56,** 1 (1989) 48–54.

32. A. Iványi A, Z. Tóth, Construction of planar de Bruijn words, In: Peák István (Ed.), *Papers on automata and languages, X. Budapest*, Karl Marx University of Economics, 1988. pp. 63–69. (ISBN: 963-7150-46-3)

33. A. Iványi, Construction of infinite de Bruijn arrays, *Discrete Applied Mathematics* **22,** 3 (1988) 289–293.

34. A. Iványi Antal, Z. Tóth, Existence of de Bruijn words, In: Gécseg F, Peák I (Ed.) *Second Conference on Automata, Languages and Programming Systems (Salgótarján, 1988).* Hungary, 1988 May 23–26. Budapest, Karl Marx University of Economics, 1988. pp. 165–172. (ISBN: 963-7150-60-9)

35. G.P. Egorychev, A. Ivanyi, A.I. Makosij, Analiz dvukh kombinatornykh summ, kharakterizuyushchikh skorost' EVM s blochnoj pamyat'yu, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **7** (1987) 19–32.

36. A. M. Iványi, A. N. Sotnikov, On the optimization of descriptor-based information retrieval systems *Vestnik Moskovskogo Universiteta Seryia 15 Vycislitelnaya Matematika i Kibernetika* **72,** 1 (1987) 51–55.

37. A. Iványi, On the d-complexity of words, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **8** (1987) 69–90.

38. A. M. Iványi, A. N., Sotnikow, On the optimization of library information retrieval systems, *Acta Cybernetica* **7,** 3 (1986) 323–328.

39. A. Iványi, I. Kátai, Modeling of priorityless processing in an interleaved memory with a perfectly informed processor, *Automation and Remote Control* **46,** 4 (1985) 520—526. Translation from *Avtom. Telemekh.* 4 (1985) 129–135.

40. A. M. Iványi, Modeling of program runs using popular Markov chains *Vestnik Moskovskogo Universiteta Seryia 15 Vycislitelnaya Matematika i Kibernetika* **3,** 1 (1984) 59–65.

41. A. M. Iványi, Estimation of the efficiency of bin-packing algorithms, *Problemy Kibernetiki* **41** (1984) 253–256.

42. A. M. Iványi, A. N. Sotnikov, Optimization of descriptor automated information retrieval systems with zone-hierarchical organization of the search set *Vestnik Moskovskogo Universiteta Seryia15 Vycislitelnaya Matematika i Kibernetika* **2** (1984) 53–57.

43. A. Iványi, Tight worst-case bounds for bin packing algorithms, In: L. Lovász, E. Szemerédi (Ed.) *Colloquium on the Theory of Algorithms,* Pécs, Hungary, 1984. July 16–21. Amsterdam, North-Holland Publishing Company, 1984. pp. 233–240. (Colloquia mathematica Societatis János Bolyai, **44**.) (ISBN: 963-8091-06-X)

44. A. Iványi, On dumpling-eating giants, In: A. Hajnal, L. Lovász, T. T. Sós (Ed.) *Finite and Infinite Sets I-II: Sixth Hungarian Combinatorial Colloquium,* Eger, Hungary, 1981.07.06-1981.07.11. Amsterdam; New York, North-Holland Publishing Company, 1984. pp. 379–390. (Colloquia Mathematica Societatis János Bolyai, **37.**) (ISBN: 963-8021-65-9; 0-444-86763-5)

45. A. M. Iványi, A. N. Sotnikov, Algorithms for determination of parameters of zone-hierarchic structures of a search set *Programmirovanie/Programming and Computer and Computer Software* **96,** 2 (1984) 68–74.

46. A. Iványi, Performance bounds for simple bin packing algorithms, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **5** (1984) 77–82.

47. A. Iványi Antal, J. Pergel, Performance evaluation of an algorithm, processing 0-1 sequences with priority *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **5** (1984) 37–40.

48. L. Hunyadvári, A. Iványi, On some complexity measures of words, *Conf. of Automata, Languages and Mathematical Systems* (Salgótarján, May 21-23, 1984.), K. Marx Univ. of Economics, Budapest, No. DM 84-2., pp. 67–82.

49. A. Iványi, J. Pergel, Parallel processing of 0-1 sequences, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **4** (1983) 85–95.

50. A. Iványi, I. Kátai, Processing of independent Markov-chains, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica* **3** (1982) 33–46.

51. A. Iványi, I. Kátai, Parallel processing of random sequences with priority, In: P. Révész, L. Schmetterer, V. M. Zolotarev (Ed.) *The First Pannonian Symposium on Mathematical Statistics.* Bad Tatzmannsdorf, Austria, 1979 Sept 16–21. New York, Springer-Verlag, 1981. pp. 122–139. (Lecture Notes in Statistics, **8.**) (ISBN: 0-387-90583-9)

52. A. Iványi, Z. Pókos, The effect of page size on the speed. *Theory of operating systems* (*Fifth Visegrád Winter School, Visegrád*, 1979). Tanulmányok – MTA Számitástech. Automat. Kutató Int. Budapest No. 100 (1979), 301–310.

53. A. Iványi, I. Kátai, Processing of random sequences with priority, *Acta Cybernetica* **4,** 1 (1978) 85–101.

54. A. Iványi Antal, I. Kátai, On the performance of computers with interleaved memory, In: Mátyás Arató, Előd Knuth (Ed.) *Selected Papers on Operating Systems: Theory and Practice (Lectures, Visegrád Winter School, Visegrád, 1978).* Budapest, Számítógép-alkalmazási Kutató Intézet (SZÁMKI), 1978. pp. 205–216.

55. A. Iványi, I. Kátai, Átfedéses memóriájú számítógépek teljsítményéről (On the performance of computers with interleaved memory), *Alkalmazott Mat. Lapok* 1-2 (1977) 1–11.

56. D. Bárdossy, A. Iványi, On some features of paging algorithms. *Theory of operating systems* (*3rd Visegrád Winter School*, Visegrád, 1977). Tanulmányok-MTA Számitástechn. Automat. Kutató Int. Budapest No. 69 (1977), 79—94.

57. A. Iványi A, I. Kátai, Estimates for speed of computers with interleaved memory systems, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Mathematica* **19** (1976) 159–164.

58. A. Iványi A, I. Kátai, On monotonic additive functions, *Acta Mathematica Academiae Scientiarum Hungaricae* **24,** 1-2 (1973) 203–208.

59. A. Iványi, On multiplicative functions with congruence property, *Annales Univ. Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Mathematica* **15** (1972) 133–137.

Conference papers

60. L. Hunyadvári, A. Iványi, On the subsequence complexity of words, *Conf. of Young Programmers and Mathematicians*, May 23–27, 1984, ELTE Budapest, pp. 7–16.

61. A. Iványi, Open questions in the algorithm performance analysis, *First Conf. of Program Designers*, July 11–12, 1985, ELTE Budapest, pp. 73–82.

62. A. Iványi, Z. Szilágyi, Analysis of the primary structure of proteins and the prognose of their secondary struture on the base of the primary one, *Second Conf. of Program Designers*, July 8–9, 1986, ELTE Budapest, pp. 53–64.

63. J. Bond, A. Iványi, Modelling of interconnection networks using De Bruijn graphs, *Third Conference of Program Designer*, ELTE, Budapest, 1987. pp. 87–88. Online access

64. A. Iványi, Construction of supercomplex matrices, *Fourth Conf. of Program Designers*, June 1–3, 1988, ELTE Budapest, pp. 133–140.

65. T. T. Cirulis, A. Iványi, On the monotonity of a "small" function, *Fourth Conf. of Program Designers*, June 1–3, 1988, ELTE Budapest, pp. 171–180.

66. A. Iványi, An algorithm for the generation of three-dimesional superperfect matrices, *Fifth Conf. of Program Designers,* Aug 28–Sept 1, 1989, ELTE Budapest, pp. 129–142.

67. A. M. Iványi, V. M. Zolotarev, Probabilistic analysis of the optimal bin packing algorithm, *Fifth Conf. of Program Designers*, Aug 28–Sept 1, 1989, ELTE Budapest, pp.183–198.

Books

68. Iványi Antal: *Párhuzamos algoritmusok*, ELTE Eötvös Kiadó, Budapest, 2003

69. A. Iványi, R. L. Smelianky, *ELements of Theoretical Programming* (in Russian), Moscow State University, 1985. 193 p.

Edited volumes

70. Iványi Antal (Ed.) *Informatikai algoritmusok: 3. Adatbázisok és alkalmazások*, Budapest: Hountler Kft., 2015. 721 p.

71. Iványi Antal (Ed.) *Informatikai algoritmusok : 2. Párhuzamos módszerek és optimalizáció*, Budapest: Hountler Kft., 2014. 810 p.

72. Iványi Antal (Ed.) *Informatikai algoritmusok 3.*, Budapest: ELTE Eötvös Kiadó, 2013. 1588 p. (ISBN: 978-963-87596-8-9)

73. Iványi Antal (Ed.) *Informatikai algoritmusok 2.* Budapest: ELTE Eötvös Kiadó, 2005. (ISBN: 963-463-775-2)

74. Iványi Antal (Ed.) *Informatikai algoritmusok 1.* Budapest: ELTE Eötvös Kiadó, 2004. 816 p. (ISBN: 963-463-664-0)

75. Iványi Antal (Ed.): *Angol-magyar informatikai szótár*, Budapest, Tinta Könyvkiadó, 2006. 390 p. (ISBN: 963 9372 79 X)

76. Donald E. Knuth: *A számítógép-programozás művészete*, 4. kötet 0. rész.: *bevezetés a kombinatorikai algoritmusokhoz és a Boole-függvényekhez* (Hungarian translation, Ed. Iványi Antal), AnTonCom Budapst, 2009. ISBN 978-963-87947-4-1

77. Donald E. Knuth: *A számítógép-programozás művészete*, 1. kötet 1. rész.: *MMIX RISC számítógép az új évezrednek* (Hungarian translation, Ed. Iványi Antal), AnTonCom Budapst, 2009. ISBN 978-963-87947-0-3

78. Donald E. Knuth: *A számítógép-programozás művészete*, 4. kötet 2. rész.: *Permutációk és* $\mathfrak{n}$*-esek előállítása* (Hungarian translation, Ed. Iványi Antal), AnTonCom Budapst, 2008. ISBN 978-963-87947-1-0

79. Donald E. Knuth: *A számítógép-programozás művészete,* 4. kötet 3. rész.: *Kombinációk és partíciók előállítása* (Hungarian translation, Ed. Iványi Antal), AnTonCom Budapst, 2008. ISBN 978-963-87947-2-7

80. Donald E. Knuth: *A számítógép-programozás művészete*, 4. kötet 4. rész.: *Fák előállítása. Kombinatorikus előállítások története* (Hungarian translation, Ed. Iványi Antal), AnTonCom Budapst, 2008. ISBN 978-963-87947-3-4

81. Antal Iványi (Ed.) *Algorithms of Informatics*. Vol. 1. Foundations. 2007. mondAT Kiadó. ISBN 978-963-87596-1-0

82. Antal Iványi (Ed.) *Algorithms of Informatics*. Vol. 2. Applications. 2007. mondAT Kiadó. ISBN 978-963-87596-2-7

83. Antal Iványi (Ed.) *Algorithms of Informatics*. Vol. 3. Selected topics 2013. Mondat Kft. ISBN 978-963-87596-7-2

84. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: *Új algoritmusok* (Hungarian translation, Ed. Iványi Antal), Scolar Könyvkiadó, 2003 ISBN 963 9193 90 93

85. Nancy Ann Lynch: *Osztott algoritmusok* (Hungarian translation, Ed. Iványi Antal), Pult Kft. 2002 ISBN: 978 963 9301030

86. T. H. Cormen, C. E. Leiserson, R. L. Rivest: *Algoritmusok* (Hungarian translation, Ed. Iványi Antal), Műszaki Könyvkiadó, 1997, 1999, 2001. ISBN 963 16 3029 3

87. Antal Iványi (Ed.): Conference of young programmers and mathematicians, ELTE, Budapest, May 23–27, 1984

88. Antal Iványi (Ed.): First conference of program designers, ELTE, Budapest, July 11–12, 1985

89. Antal Iványi (Ed.): Second conference of program designers, ELTE, Budapest, July 8–9, 1986

90. Antal Iványi (Ed.): Third conference of program designers, ELTE, Budapest, July 1–3, 1987

91. Antal Iványi (Ed.): Fourth conference of program designers, ELTE, Budapest, June 1–3, 1988

92. Antal Iványi (Ed.): Fifth conference of program designers, ELTE, Budapest, Aug 28–Sept 1, 1989

Translations

93. Knuth, Donald E.: *A számítógép-programozás művészete.* 3. kötet, Ed. Simonovits Miklós, Trans. Elekes György, Erdős Péter, Gerlits János, Hárs László, Iványi Antal, Oláh Vera, Műszaki Kiadó, Budapest, 1988, 1994.

94. V. V. Kafarov, V. L. Perov, V. P. Mesalkin: *Vegyipari rendszerek matematikai modellezése*, Trans. Iványi Antal, Nyéki György, Műszaki Könyvkiadó, Budapest, 1977.

95. G. A. Akselrud: *Tömegátadás szilárd-folyadék rendszerben,* Trans. Iványi Antal, Műszaki Könyvkiadó, Budapest, 1974.

96. A. I. Bojarinov, V. V. Kafarov: *Optimalizálás a vegyiparban*, Trans. Iványi Antal, Békássyné Molnár Erika, Sziklai Ferenc, Műszaki Könyvkiadó, Budapest, 1973. 503 pag.

97. K. F. Pavlov, P. G. Romankov, A. A. Noszkov: *Vegyipari műveletek és készülékek számítása* Ed. Podhorányi Gyula , Trans. Iványi Antal, Paal Zoltán, Pekovits László, Műszaki Könyvkiadó, Budapest, 1972. 622 pag.

DE GRUYTER OPEN
Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 17–34

DOI: 10.1515/ausi-2017-0002

On partial sorting in restricted rounds

Dedicated to the memory of Antal Iványi

Antal Iványi
Eötvös Loránd University,
Faculty of Informatics

Norbert Fogarasi
Budapest University of Technology and Economics
Department of Networked Systems and Services
email: fogarasi@hit.bme.hu

Abstract. Let n and k be integers such that $n \geq 2$ and $1 \leq k \leq n$. In this paper, we consider the problem of finding an ordered list of the k best players out of n participants by organizing a tournament of rounds of pairwise matches (comparisons). Assuming that (i) in each match there is a winner (no ties) (ii) the relative strength of the players is constant throughout the tournament and (iii) the players' strengths are transitive, the problem is equivalent to partially sorting n different, comparable objects, allowing parallelization in rounds. The rounds are restricted as one player can only play one match in each round. We propose concrete pairing algorithms and make conjectures about their performance in terms of the worst case number of rounds and matches required. The research article was started by professor Antal Iványi who sadly passed away during the work and was completed in his honor by the co-author. He hopes, in this modest way, to reflect his deep admiration for professor Iványi's many contributions to the theory, practice and appreciation of algorithm design and analysis.

Computing Classification System 1998: G.2.2.
Mathematics Subject Classification 2010: 05C85, 68R10
Key words and phrases: partial sorting, parallel sorting, tournament, pairing algorithm

1 Introduction

Sorting, that is ranking of objects using pairwise comparisons, is a common practical problem with application in different fields. Many authors describe different applications, e.g., Landau [51] biological, Hakimi [33] chemical, Kim et al. [46] and Newman et al. [57] network modelling, Bozóki, Csató, Fülöp, Kéri, Poesz, Rónyai and Temesi economical [6, 13, 14, 18, 19, 45], Liljeros et al. human relation modelling [52], while Csató, Iványi, Lucz, Móri, Pirzada, Reid, and Sótér [18, 30, 37, 38, 39, 41, 40, 43, 44, 62, 64, 65, 69] sport applications.

Partial sorting is a relaxed variant of the sorting problem in which the task is to return a list of the k largest (or k smallest) elements in order. A common practical example of partial sorting is computing the "Top 100" of some list. Martinez [56] has optimized the Quicksort algorithm for partial sorting. After unsuccessful attempts by Schreier [67] and Slupecki [68], in 1964 Kislitsyn [49] determined the number of necessary comparisons for $\mathsf{k} = 2$. Aigner [1] has proposed a general solution for $\mathsf{k} = 3$ but this has been improved by Eusterbrock [24] and Kirkpatrick [47, 48] and upper bounds have been established and proven for the required number of comparisons for $\mathsf{k} = 3$ by Hadian and Sodel [32] and Kirkpatrick [48].

Related problems are *unordered partial sorting* that is choosing the k largest elements in any order and *selection* which is choosing the k^{th} largest element of a given list. The selection problem has been extensively studied. There are many results on the lower and upper bounds of the number of necessary comparisons [10, 47, 71], near-optimal algorithms such as Ford-Johnson [28] and its improvements [7, 15, 53, 54, 55], and brute-force results [59, 60]. Finding both the largest and smallest elements at the same time has also been studied extensively [2, 3, 63, 70]. Of particular importance is the selection of the median element, for example in order to apply it as a pivot strategy for Quicksort. [22, 66, 21, 66]. The minimal and average number of necessary comparisons have been examined for median selection [32, 72, 20]. Knuth [50] remains an excellent survey of these problems and results.

With the advent of parallel computing devices, a natural direction for research is the parallelisation of sorting algorithms. Following the treatment of Pippenger [61], there are two different modes of parallelism for these problems: the case of a number of parallel comparisons equal to the number of elements which is called *balanced case*; and the case of a number of parallel comparisons large enough to allow the solution to be found in a fixed number of *rounds*: asking a fixed number of questions in the first round, processing the information then repeating this for a fixed number of rounds. This is called the *highly*

parallel case. For sorting n elements, it has long been known that, in the non-parallel case, $\Theta(n \log n)$ steps are needed. This implies that $\Theta(\log n)$ steps are needed in the balanced case, but Ajtai et al [4] showed that $\mathcal{O}(\log n)$ steps are sufficient. For the highly parallel case, Haggkvist and Hell [34] showed that $\Omega(n^{1+\frac{1}{k}})$ comparisons are needed to sort in k rounds which Alon et al [5] improved to a tighter bound. Bollobás and Thomason [11] showed that $\mathcal{O}(n^{\frac{3}{2}\log n})$ comparisons are sufficient to sort in 2 rounds which was improved by Alon et al. [5] and generalized by Bollobás and Hell [12] to $\mathcal{O}(n^{1+\frac{1}{k}\log n})$ comparisons for k rounds.

In this paper, we study the problem of partially sorting n distinct elements, in order to find the top k, in rounds, where only one pairwise comparison of each element to another is allowed in each round. We will call this a *restricted round* and is similar to sport tournaments where in each round, each team or individual can only play against one opposing team or individual. If we use the analogy of sports, we need to make the following three assumptions:

(i) in each match there is a winner (no ties)

(ii) the relative strengths of the players is constant throughout the tournament and

(iii) the players' strengths are transitive (i.e., if A beats B and B beats C then A beats C)

The task then is to come up with an optimal *pairing algorithm* which guarantees that the top k players can be found in a fixed number of rounds. Aziz et al. [8] study the related problem of determining possible and necessary winners for partially completed tournaments, Beasley et al. [9] also consider different ways of extending partial tournaments.

The structure of the paper is as follows: In section 2, formal definitions and notation are introduced, whilst in section 3 we review existing algorithms and present some related problems. In section 4, we propose some new algorithms and present some results, whilst in section 5 we draw conclusions and present directions for future research.

2 Definitions and notation

Throughout the paper, we will use the terminology of sports: the compared objects are called *players*, the comparisons *matches* and the comparisons in

each round the *pairing* of *round* i and corresponding ordering the *results* of round i. Let $n \geq 2$ be an integer denoting the total number of players and $1 \leq k \leq n$ the desired number of top players the pairing algorithm should find. In each match, the winner gets 1 point and the loser 0, so the set of permitted results is $\mathcal{R} = \{0:1, 1:0\}$. At the beginning of a tournament, each player is assigned an *index* which is an integer label $i \in \{1, \ldots, n\}$, while the *rank* of a player is the position of the player in the final ordering once the required comparisons have been made, which is equivalent to the value of the integer if we consider the equivalent problem of sorting integers $\{1, \ldots, n\}$ by pairwise comparisons.

A natural tool for the representation of the results of tournaments with n players is a directed graph G on n vertices (T_1, ..., T_n) or an $n \times n$ sized *point matrix* $\mathcal{M}$, where if player i gets $x \in \{0, 1\}$ points against player j then G contains x edges directed from T_i to T_j and $\mathcal{M}_{ij} = x$. As such, the results of tournaments can be represented by loopless directed graphs. Let $\Pi = \{\Pi_1, \ldots, \Pi_{n!}\}$ be the set of $n!$ possible permutations of the n players and $R_A(n, k, \Pi_i)$ denote the minimum required number of rounds needed for a given deterministic pairing algorithm A to rank the top k players out of n, under permutation Π_i. Our task is to find the algorithm with the least number of required rounds in the worst case, over all player permutations and the necessary number of rounds and games, i.e.,

$$R(n, k) := \min_{A \in \mathcal{A}} \max_{i} R_A(n, k, \Pi_i)\,, \tag{1}$$

where $\mathcal{A}$ is the class of all deterministic pairing algorithms which perform pairwise comparisons in rounds, pairing each player at most once in each round.

3 Existing algorithms and related problems

The two most commonly used tournament formats in sport tournaments are *round-robin* (all-play-all) and *knock-out* (elimination). Round-robin tournaments provide an upper bound for determining the full ordering, i.e., $R(n, 1) \leq R(n, 2) \leq \ldots \leq R(n, n) \leq n - 1$. Knock-out tournaments are efficient in determining the strongest player and illustrate the result that $R(n, 1) = \lceil \log_2 n \rceil$ [50]. Observing that the second best player must have been knocked out by the winner directly in one of the $\log_2 n$ rounds yields the result that $R(n, 2) = \lceil \log_2 n \rceil + \lceil \log_2 \lceil \log_2 n \rceil \rceil$. $R(n, 3)$ was first investigated by Carroll [16] who argued against the knock-out system for giving out multiple prizes

and proposed a novel algorithm to find the top three in a lawn tournament of 32 players.

A pairing algorithm which is widely used for chess tournaments with many participants (e.g., Chess Olympiad, large Open tournaments) is the *Swiss pairing system* (see [25, 26, 27]). This was first used in 1903 in the Swiss national tournament [58] although there are claims [35] that it was used in Zurich as early as in 1895. It was first formalized as an algorithm and programmed by Olafsson [58]. By construction, the Swiss pairing system has three main goals:

(i) Minimize the difference in the score of players paired against each other.

(ii) Each player should play against a new opponent in each round (unless a "bye" is requested in advance).

(iii) The same player cannot have the same colour (black or white) in three successive rounds (i.e., alternate the playing colour of each player as much as possible).

There are many variations to the original algorithm (e.g., FIDE Dubov, FIDE Dutch, FIDE Lim, FIDE Burstein, Amalfi etc. [29]) to address various shortcomings, for example the inability of the algorithm to determine the top $k \geq 2$ players and the lack of clear and widely accepted tie-breaking system for players with the same score at the end of the tournament [19, 36]. Despite this problem, to this day, the Swiss pairing system is used to give out significant monetary prizes in Open tournaments worldwide and determine the official Chess Olympiad results and medals.

Although there is a heuristic rule of thumb proposed in [35] for the required number of rounds to reliably determine the top k players out of n: $R = \frac{(n+7k)}{5}$, in 1972 Haág and Meleghegyi [31] argued in the context of a failed Hungarian National chess tournament that this is not fool-proof due to the negative incentives placed on players and the possibility of draws in chess.

Because the Swiss pairing system has different goals than the algorithm we seek, it is clear that it will not be optimal as is. However, it provides a very useful starting point and raises a number of questions which any pairing algorithm should address, such as how to pair players in the first round, or more generally large groups of players with the same score (*score group*)? How to move a player from a score group with odd number of participants to another (*floaters*)? In what order should score groups be paired?

Before we present and analyze concrete pairing algorithms, we would like to point out some related/modified pairing problems which could be analyzed. In

our current formalism, we allow up to $\lfloor \frac{n}{2} \rfloor$ matches in each restricted round. However, we could impose further restrictions to allow only a maximum of $m \leq \lfloor \frac{n}{2} \rfloor$ matches per round up to the serial case of $m = 1$. A different generalization of the problem is if a single match does not just order 2 players, but up to j of them (e.g., a horse race or swimming competition). Another complexity we may introduce to the model is that of draws in a single match which will result in the possibility of ties in the final ordering of objects. Finally, we may introduce the restrictions (ii) or (iii) from the Swiss system pairing objectives above which would make the formulation more complex, but the resulting algorithm practically more viable.

4 Newly proposed algorithms and results

4.1 Definitions and preliminary observations

At the beginning of the tournament, we assume to have no information about the relative strengths of the players, so any of them could be among the top k, thus they are all *active*. If the rank of a player is definitively determined, the player becomes *inactive*. Below, we restate some results from [42] which we will use in the construction of the proposed algorithms.

Lemma 1 *If a player has played all matches and has* l *losses then his rank is* $l + 1$ *for any* $l \in \{0, 1, \ldots, n - 1\}$.

Lemma 2 *If a player has* w *wins at any point in the tournament then his rank is at most* $n - w$.

Lemma 3 (Carroll [16]) *If a player has* k *losses at any point in the tournament then his rank is at least* $k + 1$ *and will not be among the top* k *players.*

4.2 Combined algorithm and enhancements

Based on the above results, Iványi [42] proposes an algorithm which works on the principle of the Swiss pairing algorithm (i.e., in each round it pairs players with the same or similar scores who have not yet played) and with the following two enhancements:

(i) **Transitivity rule:** At the completion of each round, once the results have been recorded in score matrix M, all of the additional results which can be deduced using the transitivity assumption are recorded.

(ii) **Deletion rule:** If the rank of a player is unambiguously determined then we delete the player and all their results from the score matrix and make them inactive.

Further enhancements can be made to the COMBINED algorithm by clarifying the following set of algorithm attributes:

1. **Ordering of score groups.** When preparing the pairings for the next round, we may begin from the top score group and move the odd player of a score group down to a lower score group (*float down*). Alternatively, we may start from the bottom score group and float up. In formally defining the Swiss pairing algorithm, Olafsson [58] argues for a bi-directional score group order: in the top half of the tournament, the odd player floats down, in the bottom half, they float up and conflicts are iteratively resolved around the middle.

2. **Ordering within a score group.** When determining the "standings" within a score group, for players of the same score, we may decide to simply apply the original indexing of players. (In chess tournaments this is usually based on the ELO rating of the players [23] and represents a pre-conceived strength order). Alternatively, the Buchholz score (sum of the scores of previous opponents) or other tie-breaking score [36] may be computed to determine an ordering within a score group.

3. **Group pairing.** Hollosi and Pahle [35] enlist four different ways in which $2m$ players with a given ordering within a score group may be paired: Fold pairing (1 vs. $2m$, 2 vs. $2m-1, \ldots, m$ vs. $m+1$), Slide pairing (1 vs. $m+1$, 2 vs. $m+2, \ldots, m$ vs. $2m$), Adjacent pairing (1 vs. 2, 3 vs. $4, \ldots, 2m-1$ vs. $2m$) or random. In particular, this method is used to determine the first round pairings for the tournament. For Swiss system chess tournaments, Slide pairing is usually applied.

4. **Handling of unpaired players.** A strict requirement of the Swiss pairing system is that all players must be paired in each round against a new opponent (unless a "bye" has been requested). However, in an optimal partial sorting algorithm this is not a necessary condition. Indeed, when players are deleted, they are omitted from further rounds. There may be pairing situations where active players must remain unpaired, otherwise existing pairings and paired score groups are disrupted. An attribute of any pairing algorithm is how it handles such situations, it may leave

players unpaired or maximize the number of paired players within a score group or combine a score group with another score group in order to maximize the number of paired players [58].

4.3 Top-Down pairing algorithm

Starting from the Combined algorithm of Iványi [42] and considering the algorithm attributes of the previous section, we propose the following Top-Down pairing algorithm. At the beginning of each round, players are sorted as follows:

(i) In increasing order by the number of losses.

(ii) If the number of losses is equal then in decreasing order by their total score.

(iii) If the total score is also equal then in decreasing order by their Buchholz score.

(iv) If all of the above are equal then in increasing order by the initial index.

Once players are sorted, we iterate in a top-down fashion, scanning down the list, trying to find an opponent for the highest ranked unpaired player. If an opponent cannot be found for a player, we leave them unpaired and move to the next unpaired player on the list. At the end of each round, we record the results in the score matrix, fill in the results implied by the transitive rule, update the Buchholz scores and determine if any player can be made inactive and possibly added to the top k players. The formal pseudocode for Top-Down, recorded in the conventions described in Cormen et al. [17] is given below.

Input. n: the total number of players; k: the number of top players to rank; $V = [V_1, V_2, \ldots, V_n]$: the relative strengths of players, a permutation of the numbers 1 to n.

Output. r: the required number of rounds; m: the required number of matches, $res = [res_1, \ldots . res_k]$: ordered list of the index of the top k elements in V.

Work variables. i, j: cycle variables; c: counter for res; M: match matrix, augmented with 7 helping metrics for each player

```
TOP-DOWN(n, k, V)
01 c = 1                          // lines 01–02: initialization of working variables
02 m = 0
03 for i = 1 to n                                  // lines 03–12: initialization of M
04     for j = 1 to n
05         M_{i,j} =null
06     M_{i,n+1} = 0                                          // total score for player i
07     M_{i,n+2} = 0                                        //number of losses for player i
08     M_{i,n+3} = 0                                         // Buchholz score for player i
09     M_{i,n+4} = 0        // flag for having been paired this round for player i
10     M_{i,n+5} = 1                                         // active flag for player i
11     M_{i,n+6} = V[i]                                              // rank for player i
12     M_{i,n+7} = i                                      // original index for player i
13 for r = 1 to n − 1                 // maximum of n − 1 rounds to be paired
14     for i = 1 to n − 1
15         if (M_{i,n+4} == 0 and M_{i,n+5} == 1)    //active and not yet paired
16             for j = i + 1 to n
17                 if (M_{j,n+4} == 0 and M_{j,n+5} == 1 and M_{i,j} == null)
18                         m = m + 1
19                         M_{j,n+4} = 1
20                         M =RecordResult(i, j, M)
21                         break
22     M =UpdateTrans(M)    // lines 22-38 tasks at the end of each round
23     M =UpdateBuchholz(M)
24     for i = 1 to n
25         M_{i,n+4} = 0                                              // reset paired flag
26     M =SortMatrix(M)                 // sort by losses, total score, Buchholz
27     for i = 1 to n − 1                 // lines 27-35 deactivate right players
28         if M_{i,n+5} == 1
29             if M_{i,n+2} < M_{i+1,n+2}
30                 res[c] = M_{i,n+7}
31                 M_{i,n+5} = 0
32                 c = c + 1
33                 if k == c − 1
34                         return r, m, res
35             else break
36     if   c == n                                    // special case for last player
37         res[c] = M_{n,n+7}
38         return r, m, res
```

Note that helper functions RecordResult, UpdateTrans, UpdateBuchhlz and SortMatrix are used in Top-Down, but their pseudocode are not listed here for the sake of brevity. They are constructed in a straightforward way as explained at the beginning of this section.

Top-Down is simple in that it does not consider score groups separately and thus it avoids the problem of floating the odd player in a score group. It is also intuitive, in that it aims to pair the strongest players against the strongest available opponent in a greedy fashion. In order to demonstrate how the algorithm works, we will present a simple example for $n = 4$. Using the notation of [42], let $P_{i,j}$ denote the player with index i and rank j. We will consdier the following permutation of four players: $\pi_1 = \{P_{1,1}, P_{2,4}, P_{3,2}, P_{4,3}\} = \{1, 4, 2, 3\}$ and assume we are interested in finding the full ordering, so $k = 4$.

The basic principle of Top-Down is that once the players are sorted (before the first round, this is done by index), the top player is paired with the highest available opponent, so $P_{1,1}$ is paired against $P_{2,4}$ and the remaining two players are paired against each other. Once the results are recorded, the stylized match matrix including relevant tournament result columns and with the round number in the index of the result is shown in Table 1.

Player	$P_{1,1}$	$P_{2,4}$	$P_{3,2}$	$P_{4,3}$	Score	Losses	Buchholz	Active
$P_{1,1}$	X	1_1			1	0	0	1
$P_{2,4}$	0_1	X			0	1	0	1
$P_{3,2}$			X	1_1	1	0	0	1
$P_{4,3}$			0_1	X	0	1	0	1

Table 1: Stylized match matrix M after the first round.

Since there are no transitive results to record, the players are then sorted in increasing order of losses, keeping the index order between tied players, yielding the ordering $P_{1,1}, P_{3,2}, P_{2,4}, P_{4,3}$. Since sorting by total score and Buchholz does not modify this order, the second round top-down pairing will be performed on this ordering. $P_{1,1}$ is now paired against $P_{3,2}$ and $P_{2,4}$ is paired against $P_{4,3}$. Table 2 shows the stylized match matrix with pre-round sorting, after the second round results are recorded, but before the application of the transitivity rule.

Applying the transitivity rule, the results of the matches $P_{1,1}$ vs. $P_{4,3}$ and $P_{3,2}$ vs. $P_{2,4}$ can be deduced and the newly sorted result matrix is shown in Table 3 with results obtained by the transitivity rule shown in bold.

Player	$P_{1,1}$	$P_{3,2}$	$P_{2,4}$	$P_{4,3}$	Score	Losses	Buchholz	Active
$P_{1,1}$	X	1_2	1_1		2	0	1	1
$P_{3,2}$	0_2	X		1_1	1	1	1	1
$P_{2,4}$	0_1		X	0_2	0	2	0	1
$P_{4,3}$		0_1	1_2	X	1	1	0	1

Table 2: Stylized match matrix M after the second round.

Player	$P_{1,1}$	$P_{3,2}$	$P_{2,4}$	$P_{4,3}$	Score	Losses	Buchholz	Active
$P_{1,1}$	X	1_2	1_1	$\mathbf{1_2}$	3	0	3	1
$P_{3,2}$	0_2	X	$\mathbf{1_2}$	1_1	2	1	1	1
$P_{4,3}$	$\mathbf{0_2}$	0_1	X	1_2	1	2	0	1
$P_{2,4}$	0_1	$\mathbf{0_2}$	0_2	X	0	3	0	1

Table 3: Stylized match matrix M after two full rounds and the transitivity rule applied.

Applying the de-activation rules of the algorithm, we can see that the full ordering has been determined in 2 rounds. Working through the same algorithm for the other 23 permutations, we observe that in one third of the cases, Top-Down finds the full ranking in 2 rounds, while in two thirds of the cases, 3 rounds are necessary, yielding an average of 2.66667 rounds to be necesesary. Some more results associated with the Top-Down algorithm for small values of n and k are presented in Table 4.

It is trivial that the cases $n = k$ and $n = k-1$ are equivalent, so the former is not even shown in the table. However, further examining the table of results, we observe that there is no difference, on average, between finding the top 2 or 3 amongst 4 players and similarly the top 6 or 7 amongst 8 players. This implies that finding the 2^{nd} best player automatically implies the 3^{rd} best out of 4 and similarly, the same round that determines the 6^{th} best always determines the 7^{th} best out of 8 players. Why these are true, but the same relationship does not hold for $n = 6$ and $k = 4$ and 5 can be the subject of future research.

Conjecture 4 *The* Top-Down *algorithm is optimal amongst deterministic algorithms in terms of the worst case number of rounds required, that is*

$$R(n,k) = \max_i \textsc{Top-Down}(n,k,\Pi_i)\,, \tag{2}$$

where $R(n,k)$ *is as defined in equation (1).*

n	k	min	max	average
2	1	1	1	$2/2 = 1$
4	1	2	2	$48/24 = 2$
4	2	2	3	$64/24 = 2.66667$
4	3	2	3	$64/24 = 2.66667$
6	1	2	3	$2040/720 = 2.83333$
6	2	2	4	$2552/720 = 3.54444$
6	3	2	5	$2808/720 = 3.9$
6	4	2	5	$2960/720 = 4.11111$
6	5	2	5	$2992/720 = 4.15556$
8	1	3	3	$120960/40320 = 3$
8	2	3	5	$160128/40320 = 3.97143$
8	3	3	6	$183296/40320 = 4.54603$
8	4	3	6	$192512/40320 = 4.77460$
8	5	3	6	$200576/40320 = 4.97460$
8	6	3	6	$206464/40320 = 5.12063$
8	7	3	6	$206464/40320 = 5.12063$

Table 4: Minimal, maximal and average number of rounds required for Top-Down for small values of n and k.

Note that we do not conjecture optimality for the number of matches, as Top-Down pairs all the players it can in each round even though in the last round this may not be necessary for the determination of the k best players.

4.4 Exhaustive pairing algorithm

In this subsection, we will assume that n is even and consider all possible, unique pairings of n elements. In the case that n is odd, we can make it even by adding a dummy element which is smaller than all other elements. Now, consider all possible pairings for the first round. After any pairing in the first round, there will be exactly $\frac{n}{2}$ winners and $\frac{n}{2}$ losers. We could then consider all possible pairings for the second round and evaluate the standings after applying the transitive and deletion rules. We could continue this in a brute-force exhaustive way and for a given input permutation of elements, find the best possible pairing which determines the top k elements most quickly.

For the trivial case of $n = 2$, a single round with the one possible pairing completes the full ranking. The following result gives a constructive solution to the exhaustive approach for all even $n \geq 4$.

Theorem 5 *For any even $n \geq 4$ there exists a pairing which, utilizing the transitivity rule, determines the full ordering of n players in exactly 2 rounds.*

Proof. Let P_i be the index of the player with rank i for all $i \in \{1, \ldots, n\}$. We first observe that the match between P_i and P_{i+1} must be played for all i, $1 \leq i \leq k-1$ because these results cannot be deduced by transitivity. Therefore, at least $k-1$ matches will be necessary to determine the top k players, or $n-1$ matches in the case $k = n$. Since at most $\frac{n}{2}$ matches can be played in one round, at least 2 rounds are needed to construct full ordering.

Now, consider the following pairing which shows that this lower bound is tight:
Round 1: $\{P_1 - P_2, P_3 - P_4, \ldots, P_{n-1} - P_n\}$. The player listed first wins each match.
Round 2: $\{P_2 - P_3, P_4 - P_5, \ldots, P_{n-2} - P_{n-1}\}$. The player listed first again wins each match.
Applying the transitivity rule, we can reconstruct the full ordering and determine the top k players $\{P_1, \ldots, P_k\}$ for any k. □

The above result demonstrates that pairing players with different scores can be efficient, if the player with the lower score wins. This observation could inspire randomly introducing such pairings into the algorithm, which would lead to stochastic pairing algorithms, but this is beyond the scope of this paper.

5 Conclusions and directions for future research

In this paper, we introduced the problem of partial sorting in restricted rounds, where in each round, each element can only be compared with at most one other element. We examined various algorithms for minimizing the number of rounds required to select the top k elements and made a conjecture about Top-Down being optimal among the class of deterministic algorithms. Further computer simulations and comparison against various benchmarks and ultimately proving this conjecture mathematically is a fertile area of future research.

We also considered exhaustively finding the best possible pairing among all possible pairings and proved that for any n and any permutation, there exists

a pairing which determines the top k elements in just 2 rounds for any k. This approach shows that pairing players with the same score is not necessarily optimal and points towards considering stochastic pairing algorithms. Sadly, professor Iványi could not complete this, his last project, so this paper is dedicated to his memory.

Acknowledgements

The authors would like to thank Gyula Császár, Matej Jusup, László Csató and Ervin Haág for useful consultations and discussions.

References

[1] M. Aigner, Selecting the top three elements, *Discrete Appl. Math.*, **4** (1982) 242–262. ⇒18

[2] M. Aigner, The double selection problem, *Discrete Math.*, **73** (1989) 3–12. ⇒18

[3] M. Aigner, Finding the maximum and minimum, *Discrete Appl. Math.*, **97** (1997) 1–12. ⇒18

[4] M. Ajtai, J. Komlos, E. Szemeredi, Sorting in $\mathcal{O}(\log \mathtt{n})$ steps, *Combinatorica*, **3,** 1 (1983) 1–19. ⇒19

[5] N. Alon, Y. Azar, U. Vishkin, Tight complexity bounds for parallel comparison sorting, *IEEE Symp. Found. Comp. Sci.*, **27** (1986) 502–510. ⇒19

[6] M. Anholcer, V. Babiy, S. Bozóki, W. W. Koczkodaj, A simplified implementation of the least squares solution for pairwise comparisons matrices. *CEJOR Cent. Eur. J. Oper. Res.* **19,** 4 (2011) 439–444. ⇒18

[7] M. Ayala-Rincón, B. T. de Abreu, J. de Sequira, A variant of the Ford-Johnson algorithm that is more space efficient. *Inf. Proc. Letters*, **102,** 5 (2007) 201–207. ⇒18

[8] H. Aziz, M. Brill, F. Fischer, P. Harrenstein, J. Lang, H. G. Seedig, Possible and necessary winners of partial tournaments, in: V. Conitzer and M. Winikoff (eds.), *Proc. of 11th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (AAMAS), IFAAMAS, 2012. 8 pages. ⇒19

[9] L. B. Beasley, D. E. Brown, K. B. Reid, Extending partial tournaments, *Math. Comput. Modelling* **50,** 1 (2009) 287–291. ⇒19

[10] M. Blum, R. W. Floyd, W. Pratt, R. L. Rivest, R. E. Tarjan, Time bounds for selection, *J. Computer System Sci.*, **7** (1973) 464–471. ⇒18

[11] B. Bollobás, A. Thomason, Parallel sorting, *Discrete Appl. Math.*, **6,** 1 (1983) 1–11 ⇒19

[12] B. Bollobás, P. Hell, Sorting and graphs, in: *Graphs and Order* (ed. I. Rival), Reidel, Boston, 1985, pp.169–184. ⇒19

[13] S. Bozóki, J. Fülöp, A. Poesz, On pairwise comparison matrices that can be made consistent by the modification of a few elements, *CEJOR Cent. Eur. J. Oper. Res.* **19** (2011) 157–175. ⇒18

[14] S. Bozóki, J. Fülöp, L. Rónyai, On optimal completion of incomplete pairwise comparison matrices, *Math. Comput. Modelling* **52** (2010) 318–333. ⇒18

[15] T. D. Bui, M. Thanh, Significant improvements to the Ford-Johnson algorithm for sorting, *BIT*, **25** (1985) 70–75. ⇒18

[16] L. Carroll, Lawn tennis tournaments, *St. James' Gazette*, August 1, 1883, 5–6. Reprinted in *The Complete Works of Lewis Carroll*, Newyork Modern Library, 1947. ⇒20, 22

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition), The MIT Press, 2009. ⇒24

[18] L. Csató, Ranking by pairwise comparisons for Swiss-system tournaments, *Cent. Eur. J. Oper. Res.*, **21,** 4 (2013) 783–803. ⇒18

[19] L. Csató, On the ranking of a Swiss system, chess team tournament, *Annals of Op. Res.,* **254,** 1–2 (2017) 17–36. ⇒18, 21

[20] W. Cunto, J. I. Munro, Average case selection, *J. ACM*, **36,** 2 (1989) 270–279. ⇒18

[21] D. Dor, J. Hástad, S. Ulfberg, U. Zwick, On lower bounds for selecting the median, *SIAM J. Discrete Math.*, **14,** 3 (2001) 299–311. ⇒18

[22] D. Dor, U. Zwick, Selecting the median, *SIAM J. Comp.*, **7**, 5 (1999) 1722–1758. ⇒18

[23] A. E. Elo, *The Rating of Chessplayers, Past and Present*, Batsford, London, 1978. ⇒23

[24] J. Eusterbrock, Errata to "Selecting the top three elements" by M. Aigner, *Discrete Appl. Math.*, **41** (1993) 131–137. ⇒18

[25] FIDE, *Handbook. 04. FIDE Swiss rules,* 2013, http://www.fide.com/component/handbook/?id=83&view=article, downloaded June 6, 2017. ⇒21

[26] FIDE, *Basic rules for Swiss Systems*, http://www.fide.com/fide/handbook.html?id=83&view=article, downloaded June 6, 2017. ⇒21

[27] FIDE, *Dutch System* https://www.fide.com/fide/handbook.html?id=167&view=article, downloaded June 6, 2017. ⇒21

[28] L. Ford, S. Johnson, A tournament problem, *Amer. Math. Monthly* **66** (1959) 387–389. ⇒18

[29] L. Forlano, *VEGA chess pairing software, User's manual* http://www.vegachess.com/tl/tl_files/music_academy/distrib/vega_en.pdf, downloaded June 6, 2017. ⇒21

[30] J. Griggs, K. B. Reid, Landau's theorem revisited, *Australas. J. Comb.* **20** (1999), 19–24. ⇒18

[31] E. Haág, Cs. Meleghegyi, A semifinal that decided nothing (Hungarian), *Magyar Sakkélet* 1972 (10), 190–191. ⇒21

[32] A. Hadian, M. Sobel, Selecting the t^{th} largest using binary errorless comparisons. TR No. 121, Univ. of Minnesota, Department of Statistics, 1969. ⇒18
[33] S. L. Hakimi, On the realizability of a set of integers as degrees of the vertices of a simple graph. *J. SIAM Appl. Math.* **10** (1962) 496–506. ⇒18
[34] R. Haggkvist, P. Hell, Parallel sorting with constant time for comparisons, *SIAM J Comput*, **10,** 3 (1981) 465–472. ⇒19
[35] A. Hollosi, M. Pahle, Swiss pairing, in *Sensei's Library,* Graz, 2013, http://senseis.xmp.net/?SwissPairing, Downloaded June 6, 2017. ⇒21, 23
[36] A. Hollosi, M. Pahle, Tie Breaker, in *Sensei's Library,* Graz, 2013, http://senseis.xmp.net/?SwissPairing, Downloaded June 6, 2017. ⇒21, 23
[37] A. Iványi, Reconstruction of complete interval tournaments, *Acta Univ. Sapientiae, Inform.*, **1,** 1 (2009) 71–88. ⇒18
[38] A. Iványi, Reconstruction of complete interval tournaments II., *Acta Univ. Sapientiae, Math.*, **2,** 1 (2010) 47–71. ⇒18
[39] A. Iványi, Directed graphs with prescribed score sequences, in: *The 7th Hungarian-Japanese Symposium on Discrete Mathematics and Applications* (ed. S. Iwata, Kyoto, May 31 - June 3, 2011), 114–123. ⇒18
[40] A. Iványi, Deciding football sequences, *Acta Univ. Sapientiae, Inform.*, **4,** 1 (2012) 130—183. ⇒18
[41] A. Iványi, Degree sequences of multigraphs. *Annales Univ. Sci. Budapest., Sect. Comp.* **37** (2012) 195–214. ⇒18
[42] A. Iványi, Z. Kása, Parallel partial ranking, *Appl. Discr. Math. and Heur. Alg.*, **1,** 3 (2015) 57–76. ⇒22, 24, 26
[43] A. Iványi, L. Lucz, T. F. Móri, P. Sótér, On the Erdős-Gallai and Havel-Hakimi algorithms. *Acta Univ. Sapientiae, Inform.* **3,** 2 (2011) 230–268. ⇒18
[44] A. Iványi, S. Pirzada, Comparison based ranking, in: *Algorithms of Informatics, Vol. 3* (ed. A. Iványi), AnTonCom, Budapest 2011, 1209–1258. ⇒18
[45] G. Kéri, On qualitatively consistent, transitive and contradictory judgment matrices emerging from multiattribute decision procedures, *CEJOR Cent. Eur. J. Oper. Res.* **19,** 2 (2011) 215–224. ⇒18
[46] H. Kim, Z. Toroczkai, I. Miklós, P. L. Erdős, L. A. Székely, Degree-based graph construction, *J. Physics*: *Math. Theor. A* **42,** 39 (2009), 392001-1-3920001.10. ⇒18
[47] D. G. Kirkpatrick, A unified lower bound for selection and set partitioning problems, *J. ACM*, **28** (1981) 150–165. ⇒18
[48] D. G. Kirkpatrick, Closing a long-standing complexity gap for selection: $V_3(42) = 50$, in *Space-efficient Data Structures, Streams, and Algorithms*, Springer Verlag, Berlin, 2013, pp.61–76 ⇒18
[49] S.S. Kislitsyn, Finding the k^{th} element in ordered set with pairwise comparisons (in Russian), Sibirsk. Mat. Zh., **2,** 5 (1964) 557–564. ⇒18
[50] D. E. Knuth, *The Art of Computer programming, Vol. 3. Sorting*, Addison-Wesley, Upper Saddle River, NJ, 1998. ⇒18, 20

[51] H. G. Landau, On dominance relations and the structure of animal societies. III. The condition for a score sequence, *Bull. Math. Biophys.* **15** (1953) 143–148. ⇒ 18

[52] F. Liljeros, C. R. Edling, L. Amaral, H. E. Stanley, Y. Aberg, The web of human sexual contacts. *Nature* **411** (2001) 907–908. ⇒18

[53] G. K. Manacher, The Ford-Johnson sorting algorithm is not optimal. *J. ACM*, **26,** 3 (1979) 441–456. ⇒18

[54] G. K. Manacher, Significant improvements to the Hwang-Lin merging algorithm, *J. ACM*, **26,** 3 (1979) 434–440. ⇒18

[55] G. K. Manacher, T. D. Bu, T. Mai, Optimal combinations of sorting and merging, *J. ACM* **36** (1989) 290–334. ⇒18

[56] C. Martínez, Partial quicksort *Proc. 6th ACM-SIAM Workshop on Algorithm Engineering and Experiments and 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics.* (2004) 5 pages. ⇒18

[57] M. Newman, A. L. Barabási, D. J. Watts, *The Structure and Dynamics of Networks.* Princeton University Press, (2006). ⇒18

[58] S. Ólaffson, Weighted matchings in chess tournaments, *J. Oper. Res. Soc.*, **41,** 1 (1990) 17–24. ⇒21, 23, 24

[59] M. Peczarski, The Ford-Johnson algorithm still unbeaten for less than 47 elements, *Inf. Proc. Letters*, **101,** 3 (2007) 126–128. ⇒18

[60] M. Peczarski, Towards optimal sorting of 16 elements. *Acta Univ. Sapientiae, Inform.* **4,** 2 (2012) 215–224. ⇒18

[61] N. Pippenger, Sorting and selecting in rounds, *SIAM J Comput*, **16,** 6 (1987) 1032–1038. ⇒18

[62] S. Pirzada, A. Iványi, Minimal digraphs with given imbalance sets, *Acta Univ. Sapientiae, Math.*, **4,** 1 (2012) 86–101. ⇒18

[63] I. Pohl, A sorting problem and its complexity, *Comm. ACM* **15,** 6 (1972) 462–464. ⇒18

[64] K. B. Reid, Tournaments: Scores, kings, generalizations and special topics, *Congr. Numer.* **115** (1996) 171–211. ⇒18

[65] K. B. Reid, C. Q. Zhang, Score sequences of semicomplete digraphs, *Bull. Inst. Combin. Appl.* **24** (1998) 27–32. ⇒18

[66] A. Schönhage, M. Paterson, N. Pippenger, Finding the median, *J. Comp. Syst. Sci.*, **13** 2 (1976) 184–199. ⇒18

[67] J. Schreier, The tournament elimination systems, *Mathesis Polska*, (7) (1932) 154–160. ⇒18

[68] L. Slupecki, On the systems of tournaments, *Colloquium Math.* **2,** 4 (1951) 286–290. ⇒18

[69] J. Temesi, L. Csató, S. Bozóki, Tennis of old times and today. An application of the partially filled comparison matrices (Hungarian), in: (ed. T. Solymosi and F. Forgó) *Balance and optimum. Case studies for the seventieth birthday of Ferenc Forgó*, Aula, Budapest, 2012, 213–245. ⇒18

[70] Á. Varecza, On the largest and smallest elements. *Ann. Univ. Sci. Budapest. Sect. Comput.*, **4** (1983) 3–10. ⇒18

[71] A. C. Yao, F. F. Yao, On the average-case complexity of selecting the k^{th} best, *SIAM J. Comp.*, **11,** 3 (1982) 428-447. ⇒18
[72] C. K. Yap, New upper bounds for selection, *Comm. ACM*, **19,** 9 (1976) 501–508. ⇒18

Received: June 12, 2017 • Revised: July 3, 2017

DE GRUYTER OPEN

Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 35–47

DOI: 10.1515/ausi-2017-0003

On multigraphic and potentially multigraphic sequences

Dedicated to the memory of Antal Iványi

Shariefuddin PIRZADA
University of Kashmir
Department of Mathematics
Srinagar, Kashmir, India
email:
pirzadasd@kashmiruniversity.ac.in

Bilal Ahmad CHAT
University of Kashmir
Srinagar, Kashmir, India
email: bilalchat99@gmail.com

Uma Tul SAMEE
Islamia College for Science and Commerce,
Srinagar, Kashmir
email: pzsamee@yahoo.co.in

Abstract. An r-graph(or a multigraph) is a loopless graph in which no two vertices are joined by more than r edges. An r-complete graph on n vertices, denoted by $K_n^{(r)}$, is an r-graph on n vertices in which each pair of vertices is joined by exactly r edges. A non-increasing sequence $\pi = (d_1, d_2, \ldots, d_n)$ of non-negative integers is said to be r-graphic if it is realizable by an r-graph on n vertices. An r-graphic sequence π is said to be potentially $S_{L,M}^{(r)}$-graphic if it has a realization containing $S_{L,M}^{(r)}$ as a subgraph. We obtain conditions for an r-graphic sequence to be potentially $S_{L,M}^{(r)}$-graphic. These are generalizations from split graphs to p-tuple r-split graph.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C07
Key words and phrases: multigraph, multigraphic sequence, potentially multigraphic sequences, split graph

1 Introduction

For a positive integer r, an r-graph(or multigraph) is a loopless graph in which no two vertices are joined by more than r edges. An r-complete graph on n vertices, denoted by $K_n^{(r)}$, is an r-graph on n vertices in which each pair of vertices is joined by exactly r edges. Clearly, $K_n^{(1)} = K_n$. A non-increasing sequence $\pi = (d_1, d_2, \ldots, d_n)$ of non-negative integers is said to be r-graphic if it is the degree sequence of an r-graph G on n vertices, and such an r-graph G is referred to as a realization of π. We take $\sigma(\pi) = \sum_{i=1}^{n} d_i$. For graph theoretical notations and definitions we refer to [9].

Let $\pi = (d_1, d_2, \ldots, d_n)$ be a non-increasing sequence of non-negative integers with $d_1 \leq \sum_{i=2}^{n} \min\{r, d_i\}$. Define $\pi_k^{'} = (d_1', d_2', \ldots, d_{n-1}')$ to be the non-increasing rearrangement of the sequence obtained from

$$(d_1, d_2, \ldots, d_{k-1}, d_{k+1}, \ldots, d_n)$$

by reducing by 1 the remaining largest terms that have not been reduced r times, and repeating the procedure d_k times. $\pi_k^{'}$ is called the residual sequence obtained from π by laying off d_k.

The following three results due to Chungphaisian [2] are generalizations from 1-graphs to r-graphs of three well-known results, one by Erdős and Gallai [3], one by Kleitman and Wang [6] and one by Fulkerson, Hoffman and Mcandrew [5].

Theorem 1 [2] *Let $\pi = (d_1, d_2, \ldots, d_n)$ be a non-increasing sequence of non-negative integers, where $\sigma(\pi)$ is even. Then π is r-graphic if and only if for each positive integer $t \leq n$,*

$$\sum_{i=1}^{t} d_i \leq rt(t-1) + \sum_{i=t+1}^{n} \min\{rt, d_i\}.$$

Theorem 2 [2] *π is r-graphic if and only if π_k' is r-graphic.*

Let the subgraph H on the vertices v_i, v_j, v_k, v_l of a multigraph G contain the edges v_iv_j and v_kv_l. The operation of deleting these edges and introducing a pair of new edges v_iv_l and v_jv_k, or v_iv_k and v_jv_l is called an elementary degree preserving transformation. If this operation is performed r times on the same edge set, it is called r-exchange.

Theorem 3 [2] *Let π be an r-graphic sequence, and let G and G' be realizations of π. Then there is a sequence of r-exchanges, $E_1, \ldots, E_k$ such that the application of these r-exchanges to G in order will result in G'.*

An r-graphic sequence π is said to be potentially $K_{m+1}^{(r)}$ if there exists a realization of π containing $K_{m+1}^{(r)}$ as a subgraph. If π has a realization G containing $K_{m+1}^{(r)}$ on the $m+1$ vertices of highest degree in G, then π is said to be potentially $A_{m+1}^{(r)}$-graphic. As a special case of Lemma 2.1 in [13], Yin showed that an r-graphic sequence is potentially $K_{m+1}^{(r)}$-graphic if and only if it is potentially $A_{m+1}^{(r)}$-graphic.

The r-join (complete product) of two r-graphs G_1 and G_2 is a graph $G = G_1 \vee G_2$ with vertex set $V(G_1) \cup V(G_2)$ and the edge set consisting of all edges of G_1 and G_2 together with the edges joining each vertex of G_1 with every vertex of G_2 by exactly r edges. Let $K_l^{(r)}$ and $K_m^{(r)}$ be complete r-graphs with l and m vertices respectively, that is the complete graphs having exactly r edges between every two vertices. The r-split graph of $K_l^{(r)}$ and $\overline{K}_m^{(r)}$ denoted by $\overline{S}_{l,m}^{(r)}$ is the graph $K_l^{(r)} \vee \overline{K_m^{(r)}}$ having $l+m$ vertices, where $\overline{K_m^{(r)}}$ (having no edges) is the complement of $K_m^{(r)}$. [14]. If π has a realization G containing $\overline{S}_{l,m}$ on the $l+m$ vertices of highest degree in G, then π is said to be potentially $\overline{A}_{l,m}$-graphic.

The following two results due to Yin [13] are generalizations from 1-graphs to r-graphs of two well-known results given by A. R. Rao [12].

Theorem 4 [13] *Let $n \geq l+1$ and $\pi = (d_1, d_2, \ldots, d_n)$ be an r-graphic sequence with $d_{l+1} \geq rl$. Then π is potentially $A_{l+1}^{(r)}$-graphic if and only if π_{l+1} is r-graphic.*

Theorem 5 [13] *Let $n \geq l+1$ and $\pi = (d_1, d_2, \ldots, d_n)$ be an r-graphic sequence with $d_{l+1} \geq 2rl-1$, then π is potentially $K_{l+1}^{(r)}$.*

An extremal problem for 1-graphic sequences to be potentially $K_l^{(1)}$-graphic was considered by Erdős, Jacobson and Lehel [4] and solved by Li et al. [7, 8]. Yin [13] generalized this extremal problem and the Erdős-Jacobson-Lehel conjecture from 1-graphs to r-graphs.

In 2014, the authors [10] proved the following assertion.

Theorem 6 [10] *If* G_1 *is a realization of* $\pi_1 = (d_1^1, \ldots, d_m^1)$ *containing* K_p *as a subgraph and* G_2 *is a realization of* $\pi_2 = (d_1^2, \ldots, d_n^2)$ *containing* K_q *as a subgraph, then the degree sequence* $\pi = (d_1, \ldots, d_{m+n})$ *of the join of* G_1 *and* G_2 *is potentially* K_{p+q}*-graphic.*

The following two results for simple graphs are due to Yin [14].

Theorem 7 [14] π *is potentially* $\overline{A}_{l,m}$*-graphic if and only if* π_l *is graphic.*

Theorem 8 [14] *Let* $n \geq l + m$ *and let* $\pi = (d_1, d_2, \cdots, d_n)$ *be a non-increasing graphic sequence. If* $d_{l+m} \geq 2l + m - 2$*, then* π *is potentially* $\overline{A}_{l,m}$*-graphic.*

A condition for a graphic sequence π to be potentially $K_4 - e$ graphic can be found in [11], where $K_4 - e$ is the graph obtained from the complete graph K_4 by deleting one edge e.

2 Bounds on the sum of squares of degrees of a multigraph

From the Cauchy-Schwarz inequality, we have

$$\sum_{i=1}^{n} |a_i b_i| \leq \Big(\sum_{i=1}^{n} |a_i|^2\Big)^{\frac{1}{2}} \Big(\sum_{i=1}^{n} |b_i|^2\Big)^{\frac{1}{2}},$$

Taking $a_i = d_i$ and $b_i = 1$, we have $\Big(\sum_{i=1}^{n} d_i\Big)^2 \leq n \sum_{i=1}^{n} d_i^2$ which implies $\frac{1}{n}\Big(\sum_{i=1}^{n} d_i\Big)^2 \leq \Big(\sum_{i=1}^{n} d_i^2\Big)$. From this and the hand shaking Lemma $\sum_{i=1}^{n} d_i = 2|E|$, we have $\frac{4|E|^2}{n} = \frac{1}{n}\Big(\sum_{i=1}^{n} d_i\Big)^2 \leq \sum_{i=1}^{n} d_i^2$.

Now we have the following observation, the proof is by using the same argument as in Theorem 1 of [1].

Lemma 9 *For an* r*-graph* G*,* $\sum_{i=1}^{n} d_i^2 \leq |E|\big(r(n-2) + \frac{2|E|}{n-1}\big)$.

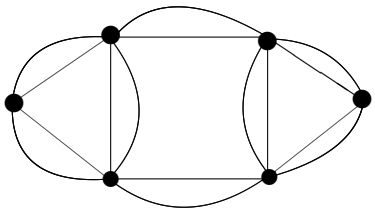

Figure 1: A 2-graph

Remark 10 *From Lemma 9, we observe that*

$$\frac{4|E|^2}{n} \leq \sum_{i=1}^{n} d_i^2 \leq |E|\big(r(n-2) + \frac{2|E|}{n-1}\big).$$

The following example shows that the equality does not hold in the above inequality.

Example 11 *Consider the 2-graph as shown in Figure 1.*

Here, $\frac{4|E|^2}{n} = \frac{4\times 16^2}{6} = \frac{512}{3} < 4^2+6^2+6^2+6^2+6^2+4^2 = 176 < 16\big(2(6-2) + \frac{2\times 16}{6-1}\big) = \frac{1152}{5}$.

Now, we have the following result.

Lemma 12 *A multigraph* G *is regular if and only if* $\frac{4|E|^2}{n} = \sum_{i=1}^{n} d_i^2$.

Proof. Suppose an r-graph G is regular of degree b. Then $2|E| = nb$ and $d_i = b$ for all $i = 1, 2, \ldots, n$. We know that $\sum_{i=1}^{n} d_i^2 = nb^2$ and $\frac{4|E|^2}{n} = \frac{1}{n}4\frac{1}{4}n^2b^2 = nb^2$. These together give $\sum_{i=1}^{n} d_i^2 = \frac{4|E|^2}{n}$.

Conversely, suppose that $\sum_{i=1}^{n} d_i^2 = \frac{4|E|^2}{n}$. Then $\frac{4}{n}|E|^2 = \sum_{i=1}^{n} d_i^2$. This implies that $\frac{1}{n}(d_1^2 + d_2^2 + \ldots + d_n^2 + 2(d_1d_2 + d_1d_3 + \ldots + d_1d_n) + \ldots + 2(d_{n-2}d_{n-1} + d_{n-2}d_n) + 2(d_{n-1}d_n)) - (d_1^2 + d_2^2 + \ldots + d_n^2) = 0$, which on simplification gives

$\frac{1}{n}\Big((d_1-d_2)^2+(d_1-d_3)^2+\ldots+(d_1-d_n)^2+(d_2-d_3)^2+(d_2-d_4)^2+\ldots+(d_2-d_n)^2+\ldots+(d_{n-1}-d_n)^2\Big)=0$. From this, we see that each term on left side is non-negative for every $i,\ j$ and right side is equal to zero. Therefore the above equation is possible when $d_i=d_j$ for every $i,\ j=1,2,\ldots,n$ and hence G is a regular graph. □

Now, we have the following observation.

Lemma 13 *Let G be an r-graph with $n>2$ vertices. Then G is a complete graph K_n^r if and only if $\frac{4|E|^2}{n}=\sum_{i=1}^{n} d_i^2=|E|\big(r(n-2)+\frac{2|E|}{n-1}\big)$.*

Proof. First we note that an r-graph G is a complete r-graph if and only if $|E|=\frac{1}{2}rn(n-1)$. Moreover, we know that $|E|=\frac{1}{2}nr(n-1)$, which implies that $2|E|(n-2)+2|E|n=nr(n-1)(n-2)+2|E|n$ and on simplication gives $\frac{4|E|^2}{n}=|E|\big(r(n-2)+\frac{2|E|}{n-1}\big)$. Thus the result follows. □

The following result partially answers the question raised in Remark 10.

Theorem 14 *A bipartite multigraph $G=K_{l,m}^{(r)}$, where $m>1$, is an r-star graph $K_{1,n-1}^{(r)}$ if and only if $\sum_{i=1}^{n} d_i^2=|E|\big(r(n-2)+\frac{2|E|}{n-1}\big)$.*

Proof. Let $K_{l,m}^{(r)}$ be an r-complete bipartite graph, where $m>1,\ n=l+m$ and $|E|=rlm$. There are l-vertices each of whose degree is $r\times m$ and m vertices each of whose degree is $r\times l$, so $\sum_{i=1}^{n} d_i^2=l(rm)^2+m(rl)^2=lr^2m^2+mr^2l^2=r^2(lm^2+ml^2)=r^2lm(l+m)$. Therefore, we have $|E|\big(r(n-2)+\frac{2|E|}{n-1}\big)=rlm\big(r(l+m-2)+\frac{2rlm}{n-1}\big)=r^2lm\big(\frac{l^2+m^2+4lm-3l-3m+2}{l+m-1}\big)$. Therefore, $r^2lm(l+m)=r^2lm\big(\frac{l^2+m^2+4lm-3l-3m+2}{l+m-1}\big)$, which gives $l=1$. Hence the result follows. □

3 Potentially r-graphic sequences

Definition 15 *Let $\overline{S}_{r_1,s_1}^{(r)},\overline{S}_{r_2,s_2}^{(r)},\ldots,\overline{S}_{r_p,s_p}^{(r)}$ be r-split graphs, respectively with r_1+s_1, r_2+s_2,..., r_p+s_p vertices. Let $L=\sum_{i=1}^{p} r_i$ and $M=\sum_{i=1}^{p} s_i$. Then*

the p*-tuple* r*-split graph, denoted by* $S^{(r)}_{L,M}$*, is the graph*

$$S^{(r)}_{L,M} = S^{(r)}_{\sum_{i=1}^{p} r_i,\ \sum_{i=1}^{p} s_i} = \overline{S}^{(r)}_{r_1,s_1} \vee \overline{S}^{(r)}_{r_2,s_2} \vee \ldots \vee \overline{S}^{(r)}_{r_p,s_p}.$$

Clearly $S^{r}_{L,M}$ has vertex set $\bigcup_{i=1}^{p} V(\overline{S}^{(r)}_{r_i,s_i})$ and the edge set consists of all edges of $\overline{S}^{(r)}_{r_1,s_1}, \overline{S}^{(r)}_{r_2,s_2}, \ldots, \overline{S}^{(r)}_{r_p,s_p}$ together with the edges joining each vertex of $\overline{S}^{(r)}_{r_i,s_i}$ with every vertex of $\overline{S}^{(r)}_{r_j,s_j}$ by exactly r-edges for every i, j with $i \neq j$.

An r-graphic sequence π is said to be potentially $S^{(r)}_{L+M}$-graphic if there exists a realization of π containing $S^{(r)}_{L+M}$ as a subgraph. If π has a realization G containing $S^{(r)}_{L+M}$ on the $L+M$ vertices of highest degree in G, then π is said to be potentially $A^{(r)}_{L+M}$-graphic.

Let $n \geq L+M$ and let $\pi = (d_1, \ldots, d_n)$ be a non-increasing sequence of non-negative integers with $d_L \geq r(L+M)-1$ and $d_{L+M} \geq rL$. We define sequences $\pi_1, \ldots, \pi_L$ as follows. Construct the sequence

$$\pi_1 = (d_2 - r, \ldots, d_L - r, d_{L+1} - r, \ldots, d_{L+M} - r, d^1_{L+M+1}, \ldots, d^1_n)$$

from π by reducing 1 from the largest term that have not been already reduced r times, and then reordering the last $n-L-M$ terms to be non-increasing. For $2 \leq i \leq r$, construct

$$\pi_i = (d_{i+1} - ir,\ \ldots,\ d_L - ir, d_{L+1} - ir,\ \ldots, d_{L+M} - ir,\ d^i_{L+M+1},\ \ldots, d^i_n)$$

from

$$\begin{aligned}\pi_{i-1} =& (d_i - (i-1)r, \ldots, d_L - (i-1)r, d_{L+1} - (i-1)r, \ldots,\\ & d_{L+M} - (i-1)r, d^{i-1}_{L+M+1}, \ldots, d^{i-1}_n)\end{aligned}$$

by deleting $d_i - (i-1)r$, reducing the first $d_i - (i-1)r$ remaining terms of d_{i-1} by one that have not been already reduced r times, and then reordering the last $n-L-M$ terms to be non-increasing.

We start with the following lemma.

Lemma 16 *If* $\pi = (d_1, d_2, \ldots, d_n)$ *is the graphic sequence of* $S^{(r)}_{L,M}$, *then*

$$\pi = \left(\left(\sum_{i=1}^{m} r(r_i + s_i - 1) \right)^{r_j}, \left(\sum_{i=1}^{m} rr_i + \sum_{i=1,\ i\neq j}^{m} rs_i \right)^{s_j} \right), \quad \text{for } j = 1, 2, \ldots, m.$$

Proof. To prove the result we use induction on m.
For $m = 1$, the result is obviously true. For $m = 2$, we have $S^{(r)}_{\sum_{i=1}^{2} r_i, \sum_{i=1}^{2} s_i}$.
Therefore for every $i = 1, 2, \ldots, r_1$ and $i = 1, 2, 3, \ldots = r_2$ and $j = 1, 2, 3, \ldots, s_1$ and $j = 1, 2, 3, \ldots, s_2$

$$\overline{d_i} = d_i + r(r_2 + s_2) \tag{1}$$

and

$$\overline{d_j} = r(r_1 + r_2 + s_2), \tag{2}$$

where $\overline{d_i}$ and $\overline{d_j}$ are respectively the degree of $\overline{v_i^{th}}$ and $\overline{v_j^{th}}$ vertex in $S_{r_1+r_2, s_1+s_2}$ and d_i is the degree of i^{th} vertex in K_{r_1}. Equations (1) and (2) hold for every i, j. Thus the graphic sequence π^2 of $S_{r_1+r_2,\ s_1+s_2}$ is

$$\begin{aligned}\pi^2 &= \left(\left(r(r_1 + s_1 - 1) + r(r_2 + s_2 \right)^{r_1}, \left(r(r_1 + s_1 - 1) + r(r_2 + s_2 \right)^{r_2}, \right. \\ &\quad \left. \left(r(r_1 + r_2 + s_2) \right)^{s_1}, \left(r(r_1 + r_2 + s_2) \right)^{s_2} \right) \\ &= \left(\left(\sum_{i=1}^{2} r(r_i + s_i - 1) \right)^{r_j}, \left(\sum_{i=1}^{m} rr_i + \sum_{i=1,\ i\neq j}^{m} rs_i \right)^{r_j} \right), \quad \text{for } j = 1, 2.\end{aligned}$$

This shows that the result is true for $m = 2$. Assume that the result holds for $m = k - 1$, therefore for all $j = 1,\ 2,\ \cdots,\ k - 1$,

$$\pi^{k-1} = \left(\left(\sum_{i=1}^{k-1} r(r_i + s_i - 1) \right)^{r_j}, \left(\sum_{i=1}^{k-1} rr_i + \sum_{i=1,\ i\neq j}^{k-1} rs_i \right)^{r_j} \right), \quad \text{for } j = 1, 2.$$

Now for $m = k$,

$$\begin{aligned} G &= S^{(r)}_{r_1,s_1} \vee S^{(r)}_{r_2,s_2} \vee \ldots \vee S^{(r)}_{r_{k-1},s_{k-1}} \vee S^{(r)}_{r_k,s_k} \\ &= A \vee S^{(r)}_{r_k,s_k}, \qquad \text{where} \quad A = S^{(r)}_{r_1,s_1} \vee S^{(r)}_{r_2,s_2} \vee \ldots \vee S^{(r)}_{r_{k-1},s_{k-1}}. \end{aligned}$$

Since the result is proved for all $m = k-1$ and using the fact that the result is proved for each pair and since the result is already proved for $k = 2$, it follows by induction hypothesis that result holds for $m = k$ also. That is,

$$\pi = \left(\left(\sum_{i=1}^{k} r(r_i + s_i - 1) \right)^{r_j}, \left(\sum_{i=1}^{k} rr_i + \sum_{i=1,\ i\neq j}^{k} rs_i \right)^{s_j} \right), \quad \text{for } j = 1, 2, \ldots, k$$

This proves the lemma. □

Lemma 17 *A non-increasing integer sequence* $\pi = (d_1, \ldots, d_n)$ *is potentially* $A^{(r)}_{L,M}$*-graphic if and only if it is potentially* $S^{(r)}_{L,M}$*-graphic.*

Proof. We only need to prove that if $\pi = (d_1, \ldots, d_n)$ is potentially $S^{(r)}_{L,M}$-graphic, then it is potentially $A^{(r)}_{L,M}$-graphic. We choose a realization G of π with vertex set $V(G) = \{v_1, \ldots, v_n\}$ such that $d_G(v_i) = d_i$ for $1 \leq i \leq n$, the induced r-subgraph $G[\{v_1, \ldots, v_{L+M}\}]$ of $\{v_1, \ldots, v_{L+M}\}$ in G contains $S^{(r)}_{L,M}$ as its r-subgraph and $|V(K^{(r)}_L) \cap \{v_1, \ldots, v_L\}|$ is maximum. Denote $H = G[\{v_1, \ldots, v_{L+M}\}]$. If $|V(K^{(r)}_L) \cap \{v_1, \ldots, d_L\}| = L$, that is, $V(K^{(r)}_L) = \{v_1, \ldots, v_L\}$, then π is potentially $A^{(r)}_{L,M}$-graphic. Assume that $|V(K^{(r)}_L) \cap \{v_1, \ldots, v_L\}| < L$. Then there exists $v_i \in \{v_1, \ldots, v_L\} \setminus V(K^{(r)}_L)$ and a $v_j \in V(K^{(r)}_L) \setminus \{v_1, \ldots, v_L\}$. Let $A = N_H(v_j) \setminus (\{v_i\} \cup N_H(v_i))$ and $B = N_G(v_i) \setminus (\{v_j\} \cup N_G(v_j))$. Since $d_G(v_i) \geq d_G(v_j)$, we have $|B| \geq |A|$. Let C be any subset of B such that $|C| = |A|$. Now form a new realization G' of π by a sequence of r-exchanges to the r-edges of the star centralized at v_j with end vertices in A with the non r-edges of the star centralized at v_j with end vertices in C, and by a sequence of r-exchange the r-edges of the star centralized at v_i with end vertices in C with the non r-edges of the star centralized at v_i with end vertices in A. It is easy to see that G' contains $S^{(r)}_{L,M}$ on $\{v_1, \ldots v_{L+M}\}$ so that $|V(K^{(r)}_L) \cap \{v_1, \ldots, v_L\}|$ is larger than that of G, which contradicts to the choice of G. □

We use the Havel-Hakimi procedure to test whether or not an r-graphic sequence π is potentially $A^{(r)}_{L,M}$-graphic.

Theorem 18 *For* $r \geq 1$ *and* $n \geq 1$*, an* r*-graphic sequence* $\pi = (d_1, \ldots, d_n)$ *is potentially* $A^{(r)}_{L,M}$*-graphic if and only if* π_L *is* r*-graphic.*

Proof. Assume that π is potentially $A^{(r)}_{L,M}$-graphic. Then π has a realization G with the vertex set $V(G) = \{v_1, \ldots, v_n\}$ such that $d_G(v_i) = d_i$ for $(1 \leq i \leq n)$

and G contains $S^{(r)}_{L,M}$ on the vertices $v_1, \ldots, v_{L+M}$, where $L + M \leq n$, so that $V^{(r)}(K_L) = \{v_1, \ldots, v_L\}$ and $V(\overline{K}^{(r)}_M) = \{v_{L+1}, \ldots, v_{L+M}\}$. By applying a sequence of r-exchanges to G in order we will show that there is one such realization G' such that $G' \setminus v_1$ has degree sequence π_1. If not, we may choose such a realization H of r-graphic sequence π such that the number of vertices adjacent to v_1 in $\{v_{L+M+1}, \ldots, v_{d_1+1}\}$ is maximum. Let $v_i \in \{v_{L+M+1}, \ldots, v_{d_1+1}\}$ and assume that there is no edge between v_1 and v_i and let $v_j \in \{v_{d_1+2}, \ldots, v_n\}$ and there are r edges between v_1 and v_j. We may assume that $d_i > d_j$. Hence there is a vertex $v_t, t \neq i, j$ such that there are r edges between v_i and v_t and no edge between v_j and v_t. Clearly $G = \left(H \setminus \{v_1^{(r)} v_j, v_i^{(r)} v_t\}\right) \bigcup \{v_1^{(r)} v_i, v_j^{(r)} v_t\}$ (where $v_i^{(r)} v_j$ means that there are r edges between v_i and v_j) is a realization of π such that $d_G(v_i) = d_i$ for $1 \leq i \leq n$, G contains $S^{(r)}_{L,M}$ on $v_1, \ldots, v_{L+M}$ with $V^{(r)}(K_L) = \{v_1, \ldots, v_L\}$ and $V(\overline{K}^{(r)}_M) = \{v_{L+1}, \ldots, v_{L+M}\}$ and G has the number of vertices adjacent to v_1 in $\{v_{L+M+1}, \ldots, v_{d_1+1}\}$ larger than that of H. This contradicts the choice of H. Repeating this procedure, we can see that π_i is potentially $A^{(r)}_{L-i}$-graphic successively for $i = 2, \ldots, L$. In particular, π_L is r-graphic.

Conversely, suppose that π_L is r-graphic and is realized by a graph G_L with a vertex set $V(G_L) = \{v_{L+1}, \ldots, v_n\}$ such that $d_{G_L}(v_i) = d_i$ for $L + 1 \leq i \leq n$. For $i = L, L-1 \ldots, 1$ form G_{i-1} from G_i by adding a new vertex v_i that is adjacent to each of $v_{i+1}, \ldots, v_{L+M}$ with r-edges and also to the vertices of G_i with degrees $s^{i-1}_{L+M+1} - r, \ldots, d^{i-1}_{d_i+1} - r$. Then for each i, G_i has degrees given by π_i and G_i contains $S^{(r)}_{L-i,M}$ on $L+M-i$ vertices $v_{i+1}, \ldots, v_{L+M}$ whose degrees are $d_{i+1} - ir, \ldots, d_{L+M} - ir$ so that $V(K^{(r)}_{L-i}) = \{v_{i+1}, \ldots, v_L\}$ and $V(\overline{K}^{(r)}_M) = \{v_{L+1}, \ldots, v_{L+M}\}$. In particular, G_0 has degrees given by π and contains $S^{(r)}_{L,M}$ on $L + M$ vertices $v_1, \ldots, v_{L+M}$ whose degrees are $d_1, \ldots, d_{L+M}$ so that $V(K^{(r)}_L) = \{v_1, \ldots, v_L\}$ and $V(\overline{K}^{(r)}_M) = \{v_{L+1}, \ldots, v_{L+M}\}$. Hence the result follows. □

The following is a sufficient condition for an r-graphic sequence to be potentially $A^{(r)}_{L,M}$-graphic.

Theorem 19 *Let $n \geq L + M$ and let $\pi = (d_1, \ldots, d_n)$ be an r-graphic sequence. If $d_{L+M} \geq 2rL + rM - 2$, then π is potentially $A^{(r)}_{L,M}$-graphic.*

Proof. Let $n \geq L + M$ and let $\pi = (d_1, \ldots, d_n)$ be a non-increasing r-graphic sequence with $d_{L+M} \geq 2rL + rM - 2$. By using the argument similar

to Theorem 8, π is potentially $K_L^{(r)}$-graphic and hence by Lemma 17, $A_L^{(r)}$-graphic. Therefore, we assume that G is a realization of π with a vertex set $V(G) = (v_1, \ldots, v_n)$ such that $d_G(v_i) = d_i$, $(1 \leq i \leq n)$ and G contains $K_L^{(r)}$ on $(v_1, \ldots, v_L)$, that is, $V(K_L^{(r)}) = \{v_1, \ldots, v_L\}$ and

$$t = e_G(\{v_1, \ldots, v_{r_1}, \ldots, v_L\}, \{v_{L+1}, \ldots, v_{L+s_1}, \ldots, v_{L+M}\})$$

(that is, the number of edges between $\{v_1, \ldots, v_L\}$ and $\{v_{L+1}, \ldots, v_{L+M}\}$) is maximum. If $t = rLM + rs_1s_2 + s_j \sum_{i=1}^{j-1} rs_i$, for $j = 3, 4, \ldots, p$, then the cardinality of the edge set of $S_{L,M}^{(r)}$ is same as t and therefore G contains $S_{L,M}^{(r)}$ on the vertices $v_1, v_2, \ldots, v_{L+M}$ with $V^{(r)}(K_M) = \{v_1, v_2, \ldots, v_L\}$ and

$$V(\overline{K^{(r)}}_M) = \{v_{L+1}, v_{L+2}, \ldots, v_{L+s_1}, \ldots, v_{L+M}\}.$$

In other-words, π is potentially $\overline{A}_{L,M}^{(r)}$-graphic. Assume that $t < \{rLM + rs_1s_2 + s_j \sum_{i=1}^{j-1} rs_i\}$, for $j = 3, 4, \ldots, p$. Then there exists a $v_k \in \{v_1, v_2, \ldots, v_{s_i}\}$ and $v_m \in \{v_{s_i+1}, v_{s_i+2}, \ldots, v_{s_i+s_j}\}$, $(i \neq j)$ such that $v_k^r v_m \notin E(G)$. Let

$$A = N_{G\setminus\{v_{s_i+1}, v_{s_i+2}, \ldots, v_{s_i+s_j}\}}(v_k) \setminus N_{G\setminus\{v_1, v_2, \ldots, v_{s_i}\}}(v_m)$$

and

$$B = N_{G\setminus\{v_{s_i+1}, v_{s_i+2}, \ldots, v_{s_i+s_j}\}}(v_k) \cap N_{G\setminus\{v_1, v_2, \ldots, v_{s_i}\}}(v_m).$$

Then $e_G(x, y) = r$ for $x \in N_{G\setminus\{v_1, \ldots, v_L\}}(v_m)$ and $y \in N_{G\setminus\{v_1, \ldots, v_{L+M}\}}(v_k)$. Otherwise, if $e_G(x, y) < r$, then $G' = (G \setminus \{v^{(r)}y, v_m^{(r)}x\}) \cup \{v_k^{(r)}v_m, x^{(r)}y\}$ is a realization of π and contains $\overline{S}_{L,M}^{(r)}$ on $v_1, \ldots, v_{L+M}$ with $V(K_L^{(r)}) = \{v_1, \ldots, v_L\}$ and $(\overline{K}_M^{(r)}) = \{v_{L+1}, \ldots, v_{L+M}\}$ such that

$$e_{G'}(\{v_1, \ldots, v_L\}, \{v_{L+1}, \ldots, v_{L+M}\}) > t,$$

which contradicts the choice of G. Thus B is r-complete. We consider the following cases.

Let $A = \emptyset$. Then $2rL + rM - 2 \leq d_k = d_G(v_k) < rL + rM - 1 + r|B|$, and so $|B| \geq rL$. Since each vertex in $N_{G\setminus v_1, \ldots, v_L}(v_m)$ is adjacent to each vertex in B by r edges and $|N_{G\setminus\{v_1, \ldots, v_L\}}(v_m)| \geq 2rL + rM - 2 = rL + rM - 1$. It can be easily seen that the r induced subgraph of $N_{G\setminus\{v_1, \ldots, v_L\}}(v_m) \cup \{v_m\}$ in G contains $\overline{S}_{L,M}^{(r)}$

as a subgraph. Thus π is potentially $\overline{A}^{(r)}_{L,M}$- graphic.

Let $A \neq \emptyset$. Let $a \in A$. If there are $x, y \in N_{G\setminus\{v_1,\ldots,v_L\}}(v_m)$ such that $e_G(x,y) < r$ then $G' = (G \setminus \{v_m^{(r)}x, v_m^{(r)}y, v_k^{(r)}a\}) \cup \{v_k^{(r)}v_m, a^{(r)}v_m, x^{(r)}y\}$ is a realization of π and contains $\overline{S}^{(r)}_{L,M}$ on $v_1, \ldots, v_{L+M}$ with $V(K_L^{(r)}) = \{v_1, \ldots, v_L\}$ and $V(\overline{K}_M^{(r)}) = \{v_{L+1}, \ldots, v_{L+M}\}$ such that $e_{G'}(\{v_1, \ldots, v_L\}, \{v_{L+1}, \ldots, v_{L+M}\}) > t$ which contradicts the choice of G. Thus $N_{G\setminus\{v_1,\ldots,v_L\}}(v_m)$ is r-complete. Since

$$|N_{G\setminus\{v_1,\ldots,v_L\}}(v_m)| \geq rL + rM - 1 \quad \text{and} \quad e_G(v_m, z) = r,$$

for any $z \in N_{G\setminus\{v_1,\ldots,v_L\}}(v_m)$, it is easy to see that the induced r-subgraph of $N_{G\setminus\{v_1,\ldots,v_L\}}(v_m) \cup \{v_m\}$ in G is r-complete, and so contains $\overline{S}^{(r)}_{L,M}$ as a r-subgraph. Thus π is potentially $\overline{A}^{(r)}_{L,M}$-graphic. □

Theorem 20 *If $\pi = (d_1, d_2, \ldots, d_n)$ is an r-graphic sequence such that $\sigma(\pi)$ is at least $(n^2 - 3n + 8)r$, then π is potentially $K_4^{(r)}$-graphic.*

Proof. Let $\pi = (d_1, d_2, \ldots, d_n)$ be an r-graphic sequence such that $d_1 \geq d_2 \geq \ldots \geq d_n \geq 1$ and $\sigma(\pi) = (n^2 - 3n + 8)r$. Suppose G is a graphical realization of π and $e(G)$ is the size of G. Then $2e(G) = \sigma(\pi)$ and $2e(G^c) = nb(n-1) - \sigma(\pi) = nr(n-1) - (n^2 - 3n + 6)r = r(2n - 6)$, so that $e(G^c) = r(n-3)$, where G^c is the complement of the r-graph G. An extremal problem is r-graph G is obtained by deleting $r(n-3)$ independent edges from the complete r-graph $K_n^{(r)}$ of order n. Hence the largest vertex number of independent sets in G^c is 3. This implies that the largest possible complete r-subgraph of G is of order 3. As $1 \leq n - 3 \leq 3$. Hence there is no complete r-subgraph of order 4 in G. Therefore, we have

$$\sigma(K_4^{(r)}, n) \geq (n^2 - 3n + 6)r + 2r = (n^2 - 3n + 8)r$$

Now Suppose that $\pi = (d_1, d_2, \ldots, d_n)$ is r-graphic sequence with $d_1 \geq d_2 \geq \ldots \geq d_n \geq r$ and $\sigma(\pi) \geq (n^2 - 3n + 8)r$. Then every graphical realization G of π is obtained by removing at most $r(n-4)$ edges from the r-complete graph $K_n^{(r)}$. Hence the maximal complete subgraph of G has order at least $n - (n-4) = 4$. Thus G is potentially $K_4^{(r)}$. In other words,

$$\sigma(K_4^{(r)}, n) \leq (n^2 - 3n + 8)r \tag{3}$$

Combining (3) and (4), the result follows. □

Acknowledgements. The authors thank the anonymous referee for his useful comments and suggestions.

References

[1] D. de Caen, An upper bound on the sum of squares of degrees in a graph, *Discrete Math.* **185** (1998) 245–248. ⇒38

[2] V. Chungphaisan, Conditions for a sequences to be r-graphic, *Discrete Math.* **7** (1974) 31–39. ⇒36, 37

[3] P. Erdős, T. Gallai, Graphs with prescribed degrees (in Hungarian) *Matematikai Lapok* **11**(1960) 264–274. ⇒36

[4] P. Erdős, M. S. Jacobson, J. Lehel, Graphs realizing the same degree sequences and their respective clique numbers, in: *Graph Theory, Combinatorics and Applications*, vol. 1, John Wiley and Sons, New York, 1991, 439–449. ⇒37

[5] D. R. Fulkerson, A. J. Hoffman, M. H. McAndrew, Some properties of graphs with multiple edges, *Canad. J. Math.* **1**7 (1965) 166–177. ⇒36

[6] D. J. Kleitman, D. L. Wang, Algorithm for constructing graphs and digraphs with given valencies and factors, *Discrete Math.* **6** (1973) 79–88. ⇒36

[7] J. S. Li, Z. X. Song, R. Luo, The Erdős-Jacobson-Lehel conjecture on potentially p_k-graphic sequences is true, *Sci. China Ser. A* **41** (1998) 510–520. ⇒37

[8] J. S. Li, Z. X. Song, An extremal problem on the potentially p_k-graphic sequence, *Discrete Math.* **212** (2000) 223–231. ⇒37

[9] S. Pirzada, *An Introduction to Graph Theory*, Universities Press, Orient Blakswan, Hyderabad, India 2012. ⇒36

[10] S. Pirzada, B. A. Chat, Potentially graphic sequences of split graphs, *Kragujevac J. Math.* **38, 1** (2014) 73–81. ⇒37, 38

[11] S. Pirzada, B. A. Chat, F. A. Dar, Graphical sequences of some family of induced subgraphs, *J. Algebra Comb. Discrete Appl.* **2(2)** (2015) 95–109. ⇒38

[12] A. R. Rao, The clique number of a graph with a given degree sequence, *Proc. Symposium on Graph Theory* (ed. A. R. Rao, Macmillan and Co. India Ltd, I.S.I. Lecture Notes Series, **4** (1979) 251–267. ⇒37

[13] J. H. Yin, Conditions for r-graphic sequences to be potentially K^r_{m+1} -graphic, *Discrete Math.* **309** (2009) 6271–6276. ⇒37

[14] J. H. Yin, A Havel-Hakimi type procedure and a sufficient condition for a sequence to be potentially $S_{r,s}$-graphic, *Czechoslovak Math. J.* **62,3** (2012) 863–867. ⇒37, 38

Received: November 30, 2016 • Revised: January 19, 2017

DE GRUYTER OPEN
Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 49–64

DOI: 10.1515/ausi-2017-0004

Parallel k_t jet clustering algorithm

Dedicated to the memory of Antal Iványi

Richárd FORSTER
Eötvös University
email: forceuse@inf.elte.hu

Ágnes FÜLÖP
Eötvös University
email: fulop@caesar.elte.hu

Abstract. The numerical simulation allows to study the high energy particle physics. It plays important of role in the reconstruction and analyze of these experimental and theoretical researches. This requires a computer background with a large capacity. Jet physics is an intensively researched area, where the factorization process can be solved by algorithmic solutions.

We studied parallelization of the k_t cluster algorithms. This method allows to know the development of particles due to the collision of high-energy nucleus-nucleus.

The Alice offline library contains the required modules to simulate the ALICE detector that is a dedicated Pb-Pb detector. Using this simulation we can generate input particles, that we can further analyzed by clustering them, reconstructing their jet structure. The FastJet toolkit is an efficient C++ implementation of the most widely used jet clustering algorithms, among them the k_t clustering. Parallelizing the standard non-optimized version of this algorithm utilizing the available CPU architecture a 1.6 times faster runtime was achieved, paving the way to drastic performance increase using many-core architectures.

Computing Classification System 1998: I.1.4
Mathematics Subject Classification 2010: 58A20
Key words and phrases: jet, cluster algorithm, database of experimenthal particle physics parallel computing, multi-core, C++11

1 Introduction

We studied the structure of the jet in the high energy physics using the many-core architecture of the modern CPUs. We consider the important concept of the particle physics.

The parton model was introduced by Richard Feynman to analyse the high-energy hadron collisions. Any hadron can be considered a composition of a number of point-like constituents.

In the theoretical physics the Quantum Chromodynamics (QCD) is the theory of strong interaction which describe the interaction between quark and gluon [5]. The QCD analogue of electric charge is a property called color. The phenomenon of color confinement, that no particle with colour charge which can be observed on its own, was introduced. So the color neutral particles are examined, then we can measure the bound states, hadrons, which are composed of quark-antiquark pair, meson or three quarks, so called baryon [6, 7].

Parton is useful for interpreting the cascades of radiation produced from QCD processes and interactions in high-energy particle collisions.

Modern CPU architectures are capable of running multiple threads at the same time, allowing the developers to utilize more resources at the same time for the same application, increasing it's performance. With the increase of complexity and performance standard tools are including more tools to ease the development for those architectures. The C++11 standard contains important tools for threading, like conditional variables and mutexes, that gives more control over the parallelized algorithms. Using these tools and applying them in a reasonable way by taking into consideration the hardware and operating system limitations a 1.6 times better performing k_t clustering was achieved without doing any additional optimizations on the code based on the FastJet libraries solution.

2 Jet in high energy physics

In the experiment the conception of jet differs from the picture which was presented by theoretical physics. We observe final state particles moving in one direction, because the information about particle origin was lost during hadronisation phase of collision. Therefore, when speaking about jets then we speak about collimated sprays of particles (Figure 1).

Therefore one should have to map a set of particle to a set of jets by special rules. These rules define a jet algorithm [8]. This method contains some

parameters, which allow us to describe the behaviours more precisely. All together they determine the definition of the jet. We apply these consideration to understand the structure of jet in the high energy experimental and theoretical physics. The results of theoretical QCD are built in the simulation, recombination and analyses of the jet research.

2.1 Jet kinematics

We introduce some quantities which are important to know in the k_t jet algorithm. The interacting partons are not generally in the centre-of-mass of colliding system, because the fraction of the hadron momentum is changing from event to event which is specified by each partons. The jets can be introduced by longitudinally boost-invariant variables because the centre-of-mass system of the partons is boosted along the direction of the colliding hadrons randomized. The mass, transverse momentum, azimutal angle and rapidity are introduced by the next expressions:

mass:	$m = \sqrt{E^2 - p_x^2 - p_y^2 - p_z^2}$
transverse momentum:	$p_T = \sqrt{p_x^2 + p_y^2}$
azimutal angle:	$\Phi = \arctan(p_y/p_x)$
rapidity:	$y = \arctan(p_x/E) = \frac{1}{2}\ln\frac{E+p_z}{E-p_z}$

In the high energy limit, when $|p| >> m$, the directly measured quantities are the following

energy E or the transverse energy: $E_T \sin\Theta \cong p_T$
the azimuth: Φ
the pseudo-rapidity: $\eta = -\ln[\tan(\Theta/2)]$,
where the polar angle is given by $\Theta = \arctan(p_T/p_z)$.

3 Jet algorithm

The origin of the basic publication of the jet finding method is Sterman and Weinberg [4] and there is huge literature which was published about the newer version of these processes [16, 15, 3, 1]. These algorithms can be divided into two major groups: cone algorithms and sequential recombination algorithms.

3.1 Cone algorithms

In the case of cone algorithm we study that particles which are situated inside the conical angular regions, then the sum of the particle's momentum concurs

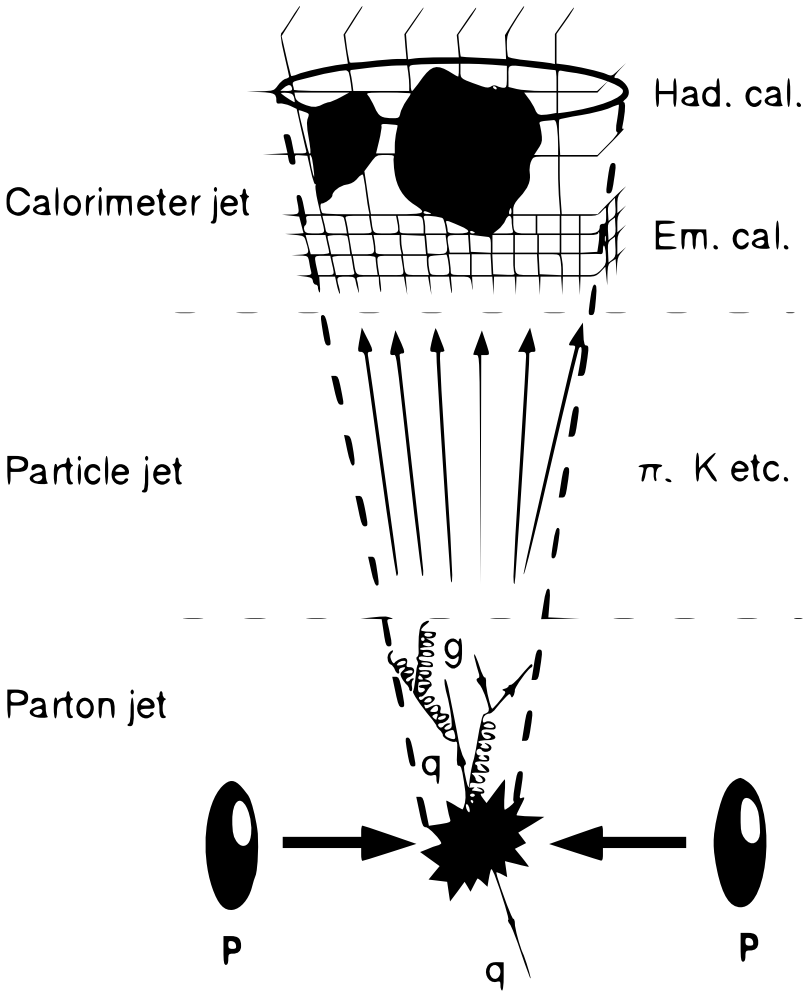

Figure 1: The structure of jet.

with the cone axis. The QCD radiation and hadronisation don't change the direction of a parton's energy flow. The stable cones are close to original partons in the direction and energy. The discrepancy between these cone algorithms are the strategy to find the stable cones and that process which is applied in that cases where the same particle is found in multiple stable cones.

3.2 Sequential recombination algorithms

This type of the algorithm identifies the closest particles in a pair to calculate the distance measure and recombine them. This process is repeat again, until it reaches a stopping criterion. The distance measure depends on the divergence of the perturbative QCD. The differences among the sequential recombination algorithms are the selecting of the distance measure and the stopping parameters.

3.2.1 The clustering algorithms

The k_t algorithm uses the final state particles in a shower which are collinear, it means that they have small transverse momentum between their constituent particles [10]. All sequential clustering algorithms have similar method. We define two distance variables.

The first of them is the one between two particles i and j, where $d_{ij} = \min\{p_{ti}^a, p_{tj}^a\} \cdot \frac{R_{ij}^2}{R}$ and a is an exponent which means the kind of the particular clustering algorithm. The value R_{ij} is determined by this expression $R_{ij}^2 = (\eta_i - \eta_j)^2 + (\Phi_i - \Phi_j)^2$ is a distance between the two particles i and j in the $(\eta - \Phi)$ space and R is the radius parameter which specifies the final size of the jet.

The second distance variable is $d_{iB} = p_{ti}^a$ this means the distance between the beam axis and the measured particle in the momentum space.

In the sequential clustering algorithms that process plays important role to find the minimum of the entire set $\{d_{ij}, d_{iB}\}$. If d_{ij} is the minimum value of the distance then particles i and j are unified into one particle (ij) and it is calculated sum of four-vectors after which i and j are removed from the list of particles. If d_{iB} is the minimum, then object i is becomed part of a final jet and removed from the list of particles.

This process is repeated until there are particles in a jet, where the distance between the axes R_{ij} greater than R, this process is an inclusive clustering. Otherwise the all amount of jets have been found, this is exclusive clustering.

k_t algorithm The a value corresponding to the k_t algorithms is 2.

The first step of the method is creating a list of the distance in the momentum-space and the distance from the beams. This algorithm involves a distance value d_{ij} between all pairs of particles i and j

$$d_{ij} = d_{ji} = min(p_{ti}^2, p_{t,j}^2)\frac{\Delta R_{ij}^2}{R^2}, \tag{1}$$

where p_{ti} means the transverse momentum of particle i with respect to the beam direction z and the expression $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\Phi_i - \Phi_j)^2$, where $y_i = \frac{1}{2}\ln\frac{E_i+p_{zi}}{E_i-p_{zi}}$ denotes the i's rapidity and Φ_i constituted the azimuth. The jet radius R is a parameter of the algorithm. This method contains a distance measure between each of particles i and the beam:

$$d_{iB} = p_{ti}^2.$$

The first kind of this method was the exclusive variation of the k_t algorithm [13], where the consideration of smallest d_{ij} and d_{iB} were introduced.

If it is a d_{ij}, we can replace i and j together with a single object which has a momentum $p_i + p_j$. This object is a pseudojet, this is neither a particle, nor a full jet.

If the smallest distance is a d_{iB}, then we take away i from the list of particles and pseudojets than we add it to the beam jet. This method is repeated until the smallest d_{ij} or d_{iB} reaches the threshold d_{cut}. All particles and pseudojets are processed.

In the case of the inclusive variation of the k_t algorithm [14] the distances d_{ij} and d_{iB} are the same as in the exclusive method.

The difference is between them when we determine the smallest value d_{iB}, then the object i is removed from the list of particle and pseudojet set and it is added to the list of final inclusive jets. There is no threshold d_{cut} and the clustering process is kept until particle or pseudojets remain.

Because the distance measures are the same in both of the inclusive and exclusive algorithms, the clustering sequence is same in these processes.

We consider the longitudinally-invariant k_t algorithm for hadron collisions [13, 14]. It was the first jet algorithm to be implemented in FastJet [9].

Longitudinally invariant k_t jet algorithm This jet method applies the inclusive version [14]. The steps of algorithm are written as following:

1. For each pair of particles i, j determine the k_t distance to use equation (1). with $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\Phi_i - \Phi_j)^2$, where p_i, y_i and Φ_i are transverse

momentum (with respect to the beam direction), rapidity and azimuth of particle i. R is a jet-radius parameter which is taken of order 1. For each parton i we calculate the beam distance $d_{iB} = p_{ti}^2$.

2. We find the minimum d_{min} of all the d_{ij}, d_{iB}. If d_{min} is a d_{ij} merge particles i and j into a single particle, then we sum their four-momenta. If it is a d_{iB} then declare particle i which is a final jet and remove it from the list.

3. Repeat from step 1 until no particle are left.

The exclusive version of the longitudinally invariant k_t jet algorithm [14] is similar, except two cases:

i. a d_{iB} is the smallest value, that particle becomes part of the beam jet.

ii. The clustering is stopped when all d_{ij} and d_{iB} are larger than the value d_{cut}.

In the next section we study the parallelization of the cluster k_t algorithm. We apply the simulation of the CERN Alice experiment offline method.

4 FastJet clustering

To do the jet clustering on the detected points, the FastJet [9] package is used. It is implemented in C++ and provides many different jet finding algorithms and analysis tools. The user can select from the widely used sequential recombination algorithms, that are implemented efficiently, while it also supports plugins for other solutions. The initial motivation to use this toolkit is the inclusion of it in the Alice Offline Framework, so after the simulation of the detector, the further process of the resulting particles can generate the jet structures.

4.1 AliRoot

AliRoot is the Off-line framework for simulation, reconstruction and analysis of the ALICE experiment at CERN. The framework and all applications are developed based on the ROOT system. Mostly it is based on Object Oriented programming paradigm, but as it was developed since 1998 it has some legacy code and other libraries, that were developed based on different principles.

The simulation part of the tool covers all processes of primary collisions and it generates the newly created particles, follows through their transportation in the detector, calculates the hits in each component. The result can be either stored in so called summable digits or digits derived from the summable ones and it can also create raw data.

For the work presented in this paper the system's TPC detected points were used. The Time Projection Chamber (TPC) detector is the main tracking component of ALICE. Particles passing through this detector ionizes the gas molecules inside the TPC and these ionization points are registered. The TPC detector consists of two cylindrical volumes sitting along the beam. These volumes are split into 18 trapezoidal readout sectors. The detector measures track positions on 159 rows [2].

4.2 Parallelizing the clustering

The goal is to not rewrite the structure of the toolkit, only include the necessary parts to utilize the parallel processing capabilities of the CPU. The used N2 clustering - which requires $O(N^2)$ operations - is considerably slower than the more optimized versions of the tile based clustering methods, but applying SIMD techniques can show rapidly how much we can gain on this field from utilizing multiple cores [17]. Even if parallelism is used, the current algorithm can lead to very inefficient solutions if not done right. It is necessary to check each element N times, which naively may result in the generation of new threads for every single particle. Even if the maximum number of threads supported by the hardware is taken into account, the overhead of creating just a few threads in each iteration results in significant bottlenecks. Thus, for such applications a generally good idea is to implement a *Thread Pool* that will create the maximum number of threads only once and keep them alive until there is any future work to do. In this initial work parallelization is done on the distance update of the newly created jets after each recombination step. This requires the new jet to being compared with other existing jets and find the nearest neighbor and to set it's distance. In case a closer jet is found the process becomes sequential on the the assignment of the new closest neighbor (Subsection 3.2.1).

4.3 Implementation

An important part of the implementation is the presence of the *Thread Pool*. The threading uses the elements of the C++11 standard, namely the pool depends on *conditional variables*, *mutexes* and *locks*. It provides two queues for storing the incoming callable functions and the input data. The required argument of the function implemented for the parallel computation is a user defined *JetData* typed parameter. This will contain the necessary values for the computation of clusters. After the pool's creation, the initialized threads

are waiting on a conditional variable until some work is presented to the FIFO. To push work into the queues and to retrieve them from there, a *unique lock* is ensuring, that only one thread can reach the container at any given time. After a job is pushed in or popped out by a worker, the lock gets released and some other threads can reach the additional if any tasks. The workers are notified through the conditional variable if they have any processing to do. After pushing in the jobs, the pool will know how much (*numberOfTasks*) work there is to do. This information is used for waiting until all the tasks are finished. This is followed by an atomic counter *done*, that is increased each time a task finishes it's work. The main thread of the application will wait on a conditional variable (*exitCondition*) of the pool, dedicated to signal the conclusion of all the processing, that have been assigned before the last *waitKernel* call. After all the work is done and the waiting is over the counters are reset to 0. At the program's end, all containers are destroyed and the threads are joined.

The function responsible for the processing of the work needs to be able to access the internal functions required for the jet clustering, so it was implemented in the *ClusterSequence class* as an additional member function. It requires a template parameter, that will tell what is the jet definition used by FastJet, when the clustering itself was instantiated.

The clusterization is done through the ClusterSequence's *_simple_N2_cluster* function, as such, this is the only routine from FastJet, that is changed. These modifications apply the currently available parallelism, creating the thread pool, preparing the work for the threads and waiting for them to be finished before moving onto the recombination step of the next jet. For the input data of the workers an evenly chunked subset of the jets are used. Because the jets are not changed, except the current one, it will not create additional race condition among the threads. The presented implementation has one part only where concurrency applies, when the new nearest neighbor is set. To prevent issues from this a *unique lock* protects the assignment of the new element.

4.3.1 Algorithm

The algorithm of the thread pool's enqueue is presented in Figure 2. The input parameter *fn* is a function pointer to the implemented worker callable *InitJetBPool* and data is a *JetData* type pointer to the input of the actual task.

The function handling the wait for finishing all running tasks is shown in Figure 3.

How the worker threads are retrieving their task and processing it is shown

```
1: procedure ENQUEUE(fn, data)
2:     lock queue_mutex
3:     tasks.push_back(fn)
4:     datas.push_back(data)
5:     ++numberOfTasks
6:     unlock queue_mutex
7: end procedure
```

Figure 2: The Enqueue function of the Thread Pool

```
1: procedure WAITALL
2:     lock wait_mutex
3:     if done != numberOfTasks then
4:         exitCondition.wait(wait_mutex)
5:         done ← 0
6:         numberOfTasks ← 0
7:     else
8:         done ← 0
9:         numberOfTasks ← 0
10:    end if
11: end procedure
```

Figure 3: The *WaitAll* function of the Thread Pool

in Figure 4. As this is the most time consuming kernel, it incorporates the shared memory to compute the triplets as fast as possible.

The routine responsible for the nearest neighbor check is described in Figure 5. The required parameters are the last jet (jetA), the new jet after jet-jet recombination (jetB), the table containing the distance between two jets (diJ), the pointer to the first element of the list containing the jets (head) and the pointer to the last element (tail). The *NN* member of jetI is the nearest neighbor of jetI, while *NN_dist* is the distance between the two. The initial NN_dist is set to R^2, where in this case R, the jet-radius parameter is set to 0.2. The parallel version of *InitJetB* also two more parameters to know which interval a specific thread needs to work on.

```
1: procedure WORKER
2:     lock queue_mutex
3:     task ← tasks.front()
4:     data ← datas.front()
5:     tasks.pop_front()
6:     datas.pop_front()
7:     unlock queue_mutex
8:     TASK(data)
9:     ++done
10:    if done == numberOfTasks then
11:        exitCondition.notify_one()
12:    end if
13: end procedure
```

Figure 4: The worker retrieving a task with it's argument and processing it

```
1: procedure InitJetB(jetA, jetB, diJ, head, tail)
2:     for jetI in head..tail do
3:         if jetI→NN == jetA or jetI→NN == jetB then
4:             find nearest neighbor for jetI
5:         end if
6:         if jetB != NULL and jetI != jetB then
7:             if distance(jetI, jetB) < jetI→NN_dist then
8:                 jetI→NN_dist = distance(jetI, jetB)
9:                 jetI→NN = jetB
10:                Update diJ accordingly
11:            end if
12:            if distance(jetI, jetB) < jetB→NN_dist then
13:                jetB→NN_dist = distance(jetI, jetB)
14:                jetB→NN = jetI
15:            end if
16:        end if
17:        if jetI→NN == tail then
18:            jetI→NN = jetA
19:        end if
20:    end for
21: end procedure
```

Figure 5: Nearest neighbor calculation after a recombination step

The part dispatching the parallel work is described in Figure 6. The required parameter is *maxThread* that tells how many concurrent threads can run on the given processor.

```
1:  procedure DispatchToPool(maxThread)
2:      JetData data[maxThread]
3:      for i in 0..maxThread do
4:          data[i].begin ← beginning of the i^th chunk of the jet list
5:          data[i].end ← end of the i^th chunk of the jet list
6:          data[i].jetA ← jetA
7:          data[i].jetB ← jetB
8:          data[i].diJ ← diJ
9:          data[i].head ← head
10:         data[i].tail ← tail
11:         ThreadPool.enqueue(InitJetBPool, data[i])
12:     end for
13:     ThreadPool.WaitAll()
14: end procedure
```

Figure 6: Dispatching work to the thread pool and synchronization at the end

4.4 Results

The complexity of the used algorithm (Subsection 4.3.1) is $O(N^2)$, which requires a high amount of computation to be done. In such case the usage of parallel computing can reduce the overall runtime. In this subsection the resulting performance increase is discussed and how the implemented thread pool helps keeping the thread creation overhead down. All tests were running on the same environment detailed in Table 1. The performance evaluation and comparison was done using the raw data from an event simulated with the AliRoot (Subsection 4.1) framework's PbPbbench test application. The detected points were directly sent to the modified FastJet routine. The generated event used for the tests contained 140535 elements.

The system used for development and testing is described in Table 1.

While applying parallelism (Subsection 4.2) on a given problem can greatly increase the performance, it also generates some overhead by creating additional threads. Comparing a parallel implementation (Subsection 4.3) with constant thread creation and another with the thread pool enabled, the threading performance of the two is far from each other. Also while using multiple threads the operating system needs to do context switching to let a specific

work be done. Figure 7 shows the runtime of the two different approach.

CPU	#Thread	OS	Compiler
Intel Core i7 4710HQ	8	Windows 10 Pro	Visual C++ 2013

Table 1: The test system

The result shows, that the naive threading based implementation took $271,31$ seconds to finish, while with the thread pool the runtime was only $207,9$ seconds, leading to a 1.3 times better performance.

Figure 8 shows the runtimes of the parallel thread pool based implementation with the original sequential clustering.

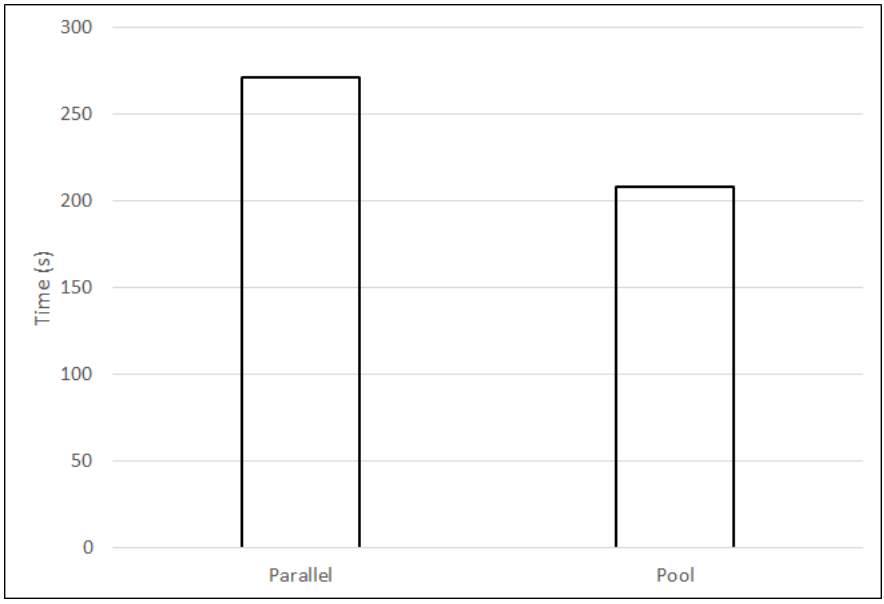

Figure 7: Runtime using naive threading or a thread pool

The parallel implementation taking $207,9$ seconds is 1.67 times faster compared to the original sequential solution's $347,18$ second long run.

Figure 9 shows the full time of the recombination loop. Comparing the parallel implementation with the performance of the optimized tile based clustering, the difference is still big.

The runtime of the parallel $O(N^2)$ algorithm is $234,52$ seconds, the tile

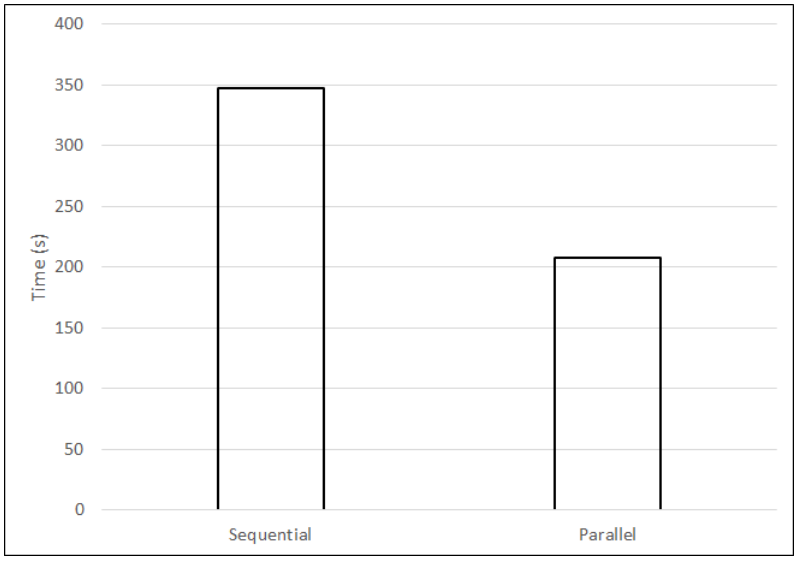

Figure 8: Runtime of the parallel $O(N^2)$ algorithm and the sequential clustering

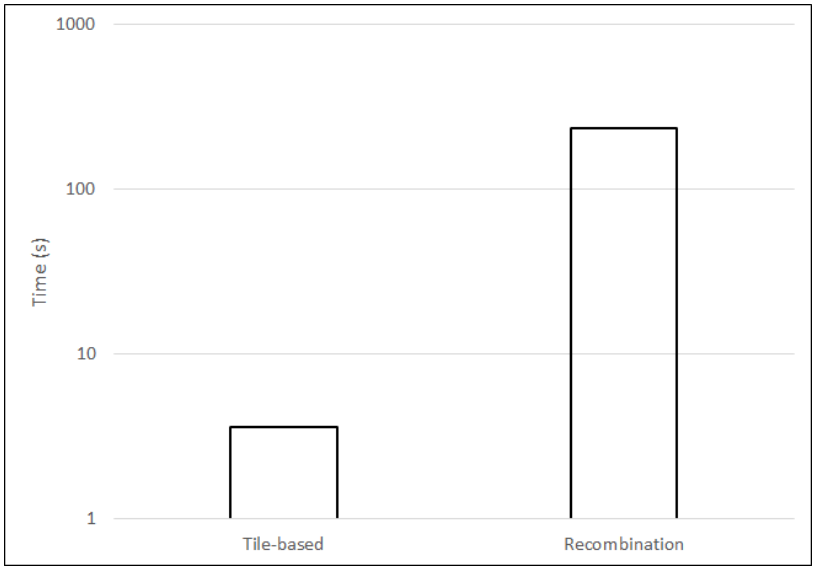

Figure 9: Runtime of the parallel $O(N^2)$ algorithm and tile based clustering

based clustering takes only $3,58$ seconds. This shows the optimized tile based clustering of the FastJet toolkit is 65.5 times faster compared to the proposed parallel method. As more work is still needs to be done on the parallelization this number can be greatly reduced giving the possibility to the parallelized algorithm to be even faster compared to the tile based solution.

5 Summary

Applying parallelization is the mean to utilize all the available processor resources independent from the given algorithm. Even if the complexity is $O(N^2)$, the performance increase is valuable. It was shown, by using a naive parallelization approach the resulting algorithm might perform better in comparison to it's sequential implementation, but the generated overhead will neglect it's positive effect. Thus by applying a thread pool on the overall system and generating the worker threads only once at the start of the application the overall runtime can be decreased considerably. Comparing the proposed parallel method to an already optimized, yet not multi-threaded, tile based clustering method also implemented in the FastJet toolkit (Section 4), the proposed algorithm still shows much slower runtime because of it's $O(N^2)$ nature.

6 Future work

To further increase the performance of the algorithm and even if not to make an $O(N^2)$ clustering faster than a tile based one, additional techniques should be explored for further optimizations and performance gain. Modern CPUs have some form of vectorization support for multiple generations now that can further speed up the evaluation of the different algorithms. This requires the data to be restructured to conform the requirements of the vectorization.

By using many-core architectures, like GPUs, it is possible to achieve full parallelization [11, 12], meaning to do all available computation in parallel. The downfall of this approach might come from the increasing amount of required memory. To fully parallelize an $O(N^2)$ algorithm, it will take N^2 memory too, which leads to impossible requirements fairly soon. It needs to be explored where the balance is and where the limit should be drawn between runtime and resource requirement to be able to run the application much faster, without needing insane amount of memory.

Furthermore it is important to not just increase the performance of the selected algorithm, but to apply the techniques and conclusions shown in this paper on other already optimized routines, to see how the overall jet clustering can benefit from parallelization.

References

[1] A. Ali, G. Kramer, Jets and QCD: A historical review of the discovery of the quark and gluon jets and its impact on QCD, *Eur. Phys. J.* H**36** (2011) 245–326. arXiv:1012.2288 [hep-ph]. ⇒51

[2] D. Rohr, S. Gorbunov, A. Szostak, M. Kretz, T. Kollegger, T. Breitner, T. Alt, ALICE HLT TPC tracking of Pb-Pb events on GPUs, *Journal of Physics: Conference Series* 396 (2012), doi:10.1088/1742-6596/396/1/012044 ⇒56

[3] G. P. Salam, Towards jetography *Eur. Phys. J.* C**67** (2010) 637-686 arXiv:0906.1833 [hep-ph]. ⇒51

[4] G. Sterman and S. Weinberg, Jets from quantum chromodynamics, *Phys. Rev. Lett.* **39** (1977) 1436. ⇒51

[5] M. E. Peskin, D. V. Schroeder, *Quantum Field Theory*, Westview Press, 1995. ⇒50

[6] T. Muta, *Foundation of Quantum Chrodinamics*, World Scientific Press 1986. ⇒50

[7] M.G. Bowler, *Femptophysics*, Pergamon Press 1990. ⇒50

[8] S. Salur, Full jet reconstruction in heavy ion collisions, *Nuclear Physics A* 830 (1-4) (2009)139c–146c. ⇒50

[9] M. Cacciari, G. P. Salam, G. Soyez, FastJet user manual, *Eur. Phys. J. C* **72** (2012) 1896 arXiv:1111.6097v1. ⇒54, 55

[10] R. Atkin, Review of jet reconstruction algorithms, *Journ. of Phys.: Conf. Ser.* **645**(2015) 012008. ⇒53

[11] R. Forster, A. Fülöp, Yang-Mills lattice on CUDA, *Acta Univ. Sapientiae, Informatica*, **5**, 2 (2013) 184–211. ⇒63

[12] R. Forster, A. Fülöp, Jet browser model accelerated by GPUs, *Acta Univ. Sapientiae Informatica* **8** 2 (2016) 171–185. ⇒63

[13] S. Carani, Yu.L Dokshitzer, M.H. Seymour, B.R. Webher, Longitudinally-invariant $k_\perp$-clustering algorithms for hadron-hadron collisions, *Nuclear Physics B* **406** (1993) 187–224. ⇒54

[14] S.D. Ellis, D. E. Soper, Successive combination jet algorithm for hadron collisions,*Phys. Rev. D* **48** 7 (1993) 3160. ⇒54, 55

[15] S. D. Ellis, J. Huston, K. Hatakeyama, P. Loch, M. Tonnesmann, Jets in Hadron-Hadron Collisions *Prog. Part. Nucl. Phys.* **60** (2008) 484 arXiv:0712.2447 [hep-ph]. ⇒51

[16] S. Moretti, L. Lönnblad and T. Sjöstrand, New and Old Jet Clustering Algorithms for Electron-Positron Events *JHEP* **9808** (1998) 001 arXiv:hep-ph/9804296. ⇒51

[17] Technology Insight: *Intel Next Generation Microarchitecture Code Name Haswell*, IDF2012. ⇒56

Received: April 3, 2017 • Revised: June 30, 2017

DE GRUYTER OPEN
Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 65–82

DOI: 10.1515/ausi-2017-0005

A unified approach of program verification

Dedicated to the memory of Antal Iványi

Tibor GREGORICS
Eötvös Loránd University
Faculty of Informatics
email: gt@inf.elte.hu

Zsolt BORSI
Eötvös Loránd University
Faculty of Informatics
email: bzsr@inf.elte.hu

Abstract. The subject of this paper is a program verification method that takes into account abortion caused by partial functions in program statements. In particular, boolean expressions of various statements will be investigated that are not well-defined. For example, a loop aborts if its execution begins in a state for which the loop condition is undefined. This work considers the program constructs of nondeterministic sequential programs and also deals with the synchronization statement of parallel programs introduced by Owicki and Gries [7]. The syntax of program constructs will be reviewed and their semantics will be formally defined in such a way that they suit the relational model of programming developed at Eötvös Loránd University [3, 4]. This relational model defines the program as a set of its possible executions and also provides definition for other important programming notions like problem and solution. The proof rules of total correctness [2, 5, 8, 9, 7] will be extended by treating abortion caused by partial functions. The use of these rules will be demonstrated by means of a verification case study.

Computing Classification System 1998: F.3.1
Mathematics Subject Classification 2010: 68N30, 68Q60
Key words and phrases: verification, programming model, program constructions, correctness

1 Introduction

In mathematics, a partial function is a binary relation from X to Y that does not map every element of X to an element of Y. It is well-known that the expression a/b is not defined if the divisor b is zero. But even the subtraction $x - y$ may have no defined value, it occurs if x and y are natural numbers and x is less than y, and the expected value also should be a natural number. More precisely, $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (where $f(x - y) = x - y$) is a partial function, because not every element of the set $\mathbb{N} \times \mathbb{N}$ is in the domain of f. The value of a partial function is undefined when its argument is out of the domain of the partial function.

In programming, an attempt to divide by zero is handled in various ways depending on the programming environment. It either leads to a compile-time error or produces a catchable runtime error if it happens at runtime. In general, evaluation to an undefined value may lead to exception or undefined behaviour. Programmers encounter with expressions on a daily basis, when they are constructing statements. The most commonly used form of the assignment statement sets the value of an expression to a variable. Particularly, boolean expressions are concerned when one writes loop condition or conditions of an alternative command. The value of these expressions is not certainly defined mathematically.

There are programs that have to work without any error. To be able to reason about the correctness of such programs we need to have rigorous definition of the semantics of the language of these programs. We also need methods that allow us to verify the correctness of programs. When one provides the semantics of a program construct, the functions that are used to build the programming statement (loop condition for example) are assumed to be total functions. The verification methods also consider program descriptions where these functions are well-defined.

This paper focuses on partial functions in program descriptions. In particular, not well defined boolean expressions of various statements will be considered. They will be took into account when providing the semantics of statements. The semantics of program constructs are given in such a way that they suit an existing relational programming model. This model defines the basic concepts of programming, for example defines the program as a set of its possible executions. A verification method will be presented as well, that handles statements containing partial logical functions. Some rules of the verification method are well known, the new rules will be given along with their proof.

The rest of this paper is organized as follows. Section 2 reviews how partial functions are handled in the literature. Section 3 introduces keywords that are allowed to use to build programs we want to investigate. Next, the semantics of these elementary programs and construct are provided. The mentioned relational programming model is also presented here shortly. Then we provide verification rule for each statement that can be formed from our keywords. Section 4 presents a verification example and illustrates the use of the verification rules given in the previous section. Section 5 summarizes our approach and work.

2 Related work

Z. Manna in [10] presents a verification method for flowchart programs. A flowchart program is a diagram constructed by edges and nodes, where the nodes denote statements. $y \leftarrow g(x, y)$ stands for an assignment statement where $g(x, y)$ is a total function mapping $D_x \times D_y$ into D_y.

C. A. Hoare in [9] did not mention that the conditions of the alternative or loop constructions were total logical functions but his examples showed this.

K. R. Apt and E.-R. Olderog use total logical functions namely Boolean expressions in their work. For example, to ensure that the expressions x div y and x mod y are total they additionally stipulate $x \operatorname{div} y = 0$ and $x \bmod y = x$ for the special case of $y = 0$ [1].

D. Gries in his fundamental work on investigating program correctness states that the guards $\beta_1, \ldots \beta_n$ have to be well-defined boolean expressions to make sure that the alternative command avoids abortion. However, in his verification rules, by assuming that all boolean expressions used in program descriptions (i.e. loop conditions, guards of guarded commands) are well-defined, he eliminates this condition to make the verification rules simpler [8].

Williem-Paul de Roever et al. extend Floyd's inductive assertion method for proving sequential transition systems. In sequential transition systems edges are labelled by commands in the form of $c \rightarrow f$. In their work c has to be a total boolean condition, but partial state transformations that might lead to runtime error (for example $c \rightarrow x := 1/y$ where c is the guard of the command $x := 1/y$) are allowed to use. They present a method for proving that the execution of a given program will not apply undefined operations [11].

3 Theoretical background

3.1 Syntax

A parallel nondeterministic program (let S denote it) can be described with a finite string of symbols including the keywords **skip**, **abort**, **if**, **fi**, **while**, **do**, **od**, [,], **await**, **then**, **ta**, **parbegin**, ‖ and **parend**, which is generated by the following grammar:

$$\begin{aligned} S = & \textbf{skip} \mid \textbf{abort} \mid \underline{v} := f(\underline{v}) \mid [S_0] \mid \textbf{await } \beta \textbf{ then } S_0 \textbf{ ta} \mid \\ & S_1; S_2 \mid \textbf{if } \pi_1 \to S_1 \,\square\, \ldots \,\square\, \pi_n \to S_n \textbf{ fi} \mid \textbf{while } \pi \textbf{ do } S_0 \textbf{ od} \mid \\ & \textbf{parbegin } S_1 \parallel \cdots \parallel S_n \textbf{ parend} \end{aligned}$$

where $\underline{v}$ denotes the current variables, $\pi, \pi_1, \ldots, \pi_n, \beta$ are partial logical functions over the current state space, $S_0, S_1, \ldots, S_n$ are programs. The **skip** is the empty program (doing nothing), **abort** is the wrong program (resulting fail), the $\underline{v} := f(\underline{v})$ is the nondeterministic assignment, the $(S_1; S_2)$ is the sequential compisition, the **if** $\pi_1 \to S_1 \,\square\, \ldots \,\square\, \pi_n \to S_n$ **fi** is the nondeterministic conditional statement, the **while** π **do** S_0 **od** is the loop, the $[S_0]$ is the atomic region, the **await** β **then** S_0 **ta** is the await-statement, and **parbegin** $S_1 \parallel \cdots \parallel S_n$ **parend** is the parallel composition.

3.2 Semantics

Before the semantics of the elementary programs and the program constructions described above are shown the concept of the program must be clarified.

All concepts of our programming model as like as the concept of the program are based on the state space. The concept of the state space has already been interpreted in several ways. For many people, the state space is a model of a von Neumann type of computer, others, e.g. Dijkstra [2], associate this notion with the problem to be solved where a state is a compound of the values of the main data types. So, the program is "outside" of the state space operating on it. In our programming model, this second meaning is used. In [3], the notion of the state space is a Cartesian product of the type value set of data types. The only mistake of this obvious definition is that it imposes an order on the components. In [5, 6], this mistake has been repaired.

A program is the complex of its executions. An execution is a sequence of states. A program, by definition, can always begin, i.e. at least one execution has to start from each state. The program is nondeterministic because several

executions may start from the same state and nobody knows which execution will happen. The first state (start state) of all executions and the last, if the execution is finite, are in the so called base state space. Namely the state space can be permanently changed; the inner states of the executions may have got new components because the program can create and destroy new components (variables) during its execution, so the state space changes dynamically. Two constraints are given: all new components have to be destroyed at the termination, at the very latest, but the base variables should never be removed. The current state always contains the components of the base state space. The variables of the base state space are the base variables; the other variables are the auxiliary variables of the program. Thus the base state space is always a subspace of the current state space. The case when the execution of a program goes wrong will be denoted by a finite sequence of states where the last state is the **fail**.

A sequence can be given by the enumeration of its elements between the signs "<" and ">": $< e_1, e_2, \cdots >$. We will use the interconnection of two sequences if the end of the first sequence is identical to the front of the second one. More precisely, if $\alpha =< a_1, \ldots, a_n >$ and $\beta =< b_1, b_2, \cdots >$ are sequences and $a_n = b_1 \neq \mathbf{fail}$, then their interconnection is $\alpha \otimes \beta =< a_1, \ldots, a_n, b_2, \cdots >$.

The formal definition of the program [6] requires some notions. Let H^{**} denote the set of all finite and infinite sequences of the elements of set H. H^{∞} includes the infinite sequences; H^* contains the finite ones. So, $H^{**} = H^* \cup H^{\infty}$ and $H^* \cap H^{\infty} = \emptyset$. The length of the sequence $\alpha \in H^{**}$ is $|\alpha|$, the value of which is ∞ if the sequence is infinite.

Definition 1 *Let A be the so-called base state space and $\bar{A}$ be the set of all states which belong to the state spaces B whose subspace is A, i.e. $\bar{A} = \bigcup_{A \leq B} B$. $\bar{A}$ does not contain the state **fail**. The relation $S \subseteq A \times (\bar{A} \cup \{\mathbf{fail}\})^{**}$ is a **program** over A, if*

1. $\mathcal{D}_S = A$
2. $\forall a \in A$ *and* $\forall \alpha \in S(a) : |\alpha| \geq 1$ *and* $\alpha_1 = a_i$
3. $\forall \alpha \in \mathcal{R}_S$ *and* $\forall i (1 \leq i < |\alpha|) : \alpha_i \neq \mathbf{fail}$
4. $\forall \alpha \in \mathcal{R}_S : |\alpha| < \infty \rightarrow \alpha_{|\alpha|} \in A \cup \{\mathbf{fail}\}$

Now, we are going to give the semantics of the elementary programs and program constructions so that they are treated as programs in the sense of

the previous definition. Our aim is to define the set of state-sequences that are mapped to an arbitrary state by a construction.

Definition 2 *Let* A *be a state space and* $\sigma \in A$ *be the current state.*
$\mathbf{skip}(\sigma) \ ::= \ \{< \sigma >\}$

Definition 3 *Let* A *be a state space and* $\sigma \in A$ *be the current state.*
$\mathbf{abort}(\sigma) \ ::= \ \{< \sigma, \mathbf{fail} >\}$

Definition 4 *Let* A *be a state space and* $f \subseteq A \times A$ *be a relation and* $\sigma \in A$ *be the current state.*

$$(\underline{v} := f(\underline{v}))(\sigma) \ ::= \begin{cases} \{< \sigma, \sigma' > | \ \sigma' \in f(\sigma)\} & \text{if } \sigma \in \mathcal{D}_f \\ \{< \sigma, \mathbf{fail} >\} & \text{if } \sigma \notin \mathcal{D}_f \end{cases}$$

Definition 5 *Let* A *be the common base state space of the program* S_1 *and* S_2*. Let* $\sigma \in A$ *be the current state.*

$$\begin{aligned} (S_1; S_2)(\sigma) \ ::= & \{\alpha \mid \alpha \in S_1(\sigma) \cap \overline{A}^{\infty}\} \cup \\ & \{\alpha \mid \alpha \in S_1(\sigma) \text{ and } \mid \alpha \mid < \infty \text{ and } \alpha_{|\alpha|} = \mathbf{fail}\} \cup \\ & \{\alpha \otimes \beta \mid \alpha \in S_1(\sigma) \cap \overline{A}^{*} \text{ and } \beta \in S_2(\alpha_{|\alpha|})\} \end{aligned}$$

Definition 6 *Let* A *be the common base state space of the program* $S_1 \ldots S_n$ *and the conditions* $\pi_1 \ldots \pi_n$*. Let* $\sigma \in A$ *be the current state.*

$$(\mathbf{if}\ \pi_1 \to S_1 \ \square \ldots \square\ \pi_n \to S_n\ \mathbf{fi})(\sigma) \ ::= \ \omega(\sigma) \cup \bigcup_{\substack{i=1 \\ \sigma \in \mathcal{D}_{\pi_i} \wedge \pi_i(\sigma)}}^{n} S_i(\sigma)$$

$$\text{where } \omega(\sigma) = \begin{cases} \{< \sigma, \mathbf{fail} >\} & \text{if } \exists i \in [1..n] : \sigma \notin \mathcal{D}_{\pi_i} \vee \forall i \in [1..n] : \sigma \in \mathcal{D}_{\pi_i} \wedge \neg \pi_i(\sigma) \\ \emptyset & \text{otherwise} \end{cases}$$

Definition 7 *Let* A *be the common base state space of the program* S_0 *and the condition* π*. Let* $\sigma \in A$ *be the current state.*

$$(\mathbf{while}\ \pi\ \mathbf{do}\ S_0\ \mathbf{od})(\sigma) \ ::= \begin{cases} (S_0; \mathbf{while}\ \pi\ \mathbf{do}\ S_0\ \mathbf{od})(\sigma) & \text{if} \quad \sigma \in \mathcal{D}_\pi \wedge \pi(\sigma) \\ \{< \sigma >\} & \text{if} \quad \sigma \in \mathcal{D}_\pi \wedge \neg\pi(\sigma) \\ \{< \sigma, \mathbf{fail} >\} & \text{if} \quad \sigma \notin \mathcal{D}_\pi \end{cases}$$

Definition 8 *Let* A *be a state space and* $f \subseteq A \times A$ *be a relation and* $\sigma \in A$ *be the current state.*
$[S](\sigma) ::= S(\sigma)$

Definition 9 *Let* A *be the common base state space of the program* S_0 *and the condition* β*. Let* $\sigma \in A$ *be the current state.*

$$(\textbf{await } \beta \textbf{ then } S_0 \textbf{ ta})(\sigma) ::= \begin{cases} \textbf{abort}(\sigma) & \textit{if} \quad \sigma \notin \mathcal{D}_\beta \\ [\textbf{if } \beta \textbf{ then } S_0 \textbf{ fi }](\sigma) & \textit{if} \quad \sigma \in \mathcal{D}_\beta \wedge \beta(\sigma) \\ (\textbf{skip}; \textbf{await } \beta \textbf{ then } S_0 \textbf{ ta})(\sigma) & \textit{if} \quad \sigma \in \mathcal{D}_\beta \wedge \neg\beta(\sigma) \end{cases}$$

Before the definition of the parallel composition the concept of the "uninterrupted" must be intoduced. Let S be a program and σ be an arbitrary state of its base state space. The execution $S(\sigma)$ is uninterrupted if the program S is the **skip**, **abort**, an assignment statement, an atomic region $[P]$ or an await statement **await** β **then** S_0 **ta** where $\sigma \notin \mathcal{D}_\beta$ or $\beta(\sigma)$ is *true*.

We must remark that in case of $S(\sigma)$ is not uniterrupted (where σ is an arbitrary state and S is a program), then there is a finite set H of program pairs (u, T) so that $u(\sigma)$ is uninterrupted and $S(\sigma) = \bigcup\limits_{(u,T)\in H} (u; T)(\sigma)$. This u is named as the first statement of S and T is the remainder part of S relative to the state σ.

Definition 10 *Let* A *be the common base state space of the programs* $S_1 \dots S_n$ *and* $\sigma \in A$ *be the current state.*

$$(\textbf{parbegin } S_1 \parallel \cdots \parallel S_i \parallel \cdots \parallel S_n \textbf{ parend})(\sigma) ::= \bigcup_{i=1}^{n} B_i(\sigma)$$

where

$$B_i(\sigma) = \begin{cases} (S_i; \textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend})(\sigma) \\ \qquad \textit{if } S_i(\sigma) \textit{ is uninterrupted} \\ \bigcup\limits_{(u,T)\in H} (u; \textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel T \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend})(\sigma) \\ \qquad \textit{if } S_i(\sigma) \textit{ is not uninterrupted and } S_i(\sigma) = \bigcup\limits_{(u,T)\in H} (u; T)(\sigma) \end{cases}$$

Let S be a program, σ be an arbitrary state of its base state space and the execution $S(\sigma)$ is not uniterrupted. A finite set H of program pairs (u, T) so that $u(\sigma)$ is uninterrupted and $S(\sigma) = \bigcup\limits_{(u,T)\in H} (u; T)(\sigma)$ can be given as follows:

- If $S = (S_1; S_2)$ and $S_1(\sigma)$ is uninterrupted, then $S(\sigma) = (S_1; S_2)(\sigma)$.
  If $S = (S_1; S_2)$ where $S_1(\sigma)$ is not uninterrupted and $S_1(\sigma) = \bigcup_{(u,T)\in H} (u; T)(\sigma)$,
  then $S(\sigma) = \bigcup_{(u,T)\in H} (u; (T; S_2))(\sigma)$.

- If $S = \textbf{if } \pi_1 \to S_1 \,\square \ldots \square\, \pi_n \to S_n \textbf{ fi}$ and all of its conditions are defined and some of them are true in σ ($\forall i \in [1..n] : \sigma \in \mathcal{D}_{\pi_i} \wedge \exists i \in [1..n] : \sigma \in \pi_i(\sigma)$), then
  $S(\sigma) = \bigcup_{\substack{i=1 \\ \sigma\in\mathcal{D}_{\pi_i}\wedge\pi_i(\sigma)}}^{n} (\textbf{skip}; S_i)(\sigma)$

- If $S = \textbf{while } \pi \textbf{ do } S_0 \textbf{ od}$ and $\sigma \in \mathcal{D}_\pi \wedge \neg\pi(\sigma)$,
  then $S(\sigma) = (\textbf{skip}; (S_0; \textbf{while } \pi \textbf{ do } S_0 \textbf{ od}))(\sigma)$.

- If $S = \textbf{await } \beta \textbf{ then } S_0 \textbf{ ta}$ and $\sigma \in \mathcal{D}_\beta \wedge \neg\beta(\sigma)$,
  then $S(\sigma) = (\textbf{skip}; \textbf{await } \beta \textbf{ then } S_0 \textbf{ ta}))(\sigma)$.

- If $S = \textbf{parbegin } S_1 \parallel \cdots \parallel S_i \parallel \cdots \parallel S_n \textbf{ parend}$, then $S(\sigma) = \bigcup_{i=1}^{n} B_i(\sigma)$
  where

$$B_i(\sigma) = \begin{cases} (S_i; \textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend})(\sigma) \\ \qquad \text{if } S_i(\sigma) \text{ is uninterrupted} \\ \bigcup_{(u_{i_j}, T_{i_j})\in H_i} (u_{i_j}; \textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel T \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend})(\sigma) \\ \qquad \text{if } S_i(\sigma) \text{ is not uninterrupted and } S_i(\sigma) = \bigcup_{(u_{i_j}, T_{i_j})\in H_i} (u_{i_j}; T_{i_j})(\sigma) \end{cases}$$

3.3 Verification

Informally, a program is correct if it satisfies the intended input/output relation. Program correctness is expressed by so-called correctness formulas. These are statements of the form

$$\{\{Q\}\}S\{\{R\}\}$$

where S is a program and Q and R are assertions. The assertion Q is the precondition of the correctness formula and R is the postcondition. The precondition describes the set of initial states in which the program S is started and the postcondition describes the set of desirable final states.

More precisely: a correctness formula is true if every excecution of S that starts in a state satisfying Q is finite (it terminates) and its final state satisfies R. (This is the concept of the total correctness. The partial correctness is omitted in this paper.)

Reasoning about correctness formulas in terms of semantics is not very convenient. Hoare has introduced a proof system allowing us to prove partial correctness of deterministic programs in a syntax-directed manner, by induction on the program syntax [9]. Dijkstra, Gries and Owicki have developed this system for nondeterministic and parallel programs [2, 8, 7]. Now this system is going to be extended.

The first six rules are well-known, they are only shown for the sake of completeness without their proofs.

Theorem 11 *Let* Q *and* R *be two assertions.*

$$\frac{Q \implies R}{\{\{Q\}\}\ \mathbf{skip}\ \{\{R\}\}}$$

Theorem 12 *Let* Q *and* R *be two assertions,* $\underline{v} := f(\underline{v})$ *be an assignment.*

$$\frac{Q \implies \underline{v} \in \mathcal{D}_f \wedge \forall \underline{e} \in f(\underline{v}) : R^{\underline{v} \leftarrow \underline{e}}}{\{\{Q\}\}\ \underline{v} := f(\underline{v})\ \{\{R\}\}}$$

The $R^{\underline{v} \leftarrow \underline{e}}$ means that the components of $\underline{v}$ must be substituted for the corresponding components of $\underline{e}$. In that case when the relation $f \subseteq A \times A$ of the assingment is a total function, i.e., f is a (deterministic) function mapping from A to A and $\mathcal{D}_f = A$, the rule of assigment can be written in the following form.

$$\frac{Q \implies R^{\underline{v} \leftarrow f(\underline{v})}}{\{\{Q\}\}\ \underline{v} := f(\underline{v})\ \{\{R\}\}}$$

Theorem 13 *Let* Q *and* R *be two assertions,* S *be a program.*

$$\frac{\{\{Q\}\}\ S\ \{\{R\}\}}{\{\{Q\}\}\ [S]\ \{\{R\}\}}$$

Theorem 14 *Let* Q *and* R *be two assertions,* S_1 *and* S_2 *be two programs.*

$$\frac{\begin{array}{c}\exists Q' : A \to \mathbb{L} \\ \{\{Q\}\}\ S_1\ \{\{Q'\}\} \\ \{\{Q'\}\}\ S_2\ \{\{R\}\}\end{array}}{\{\{Q\}\}\ (S_1; S_2)\ \{\{R\}\}}$$

Theorem 15 *Let* Q *and* R *be two assertions, and* S^* *stand for the program* S *annotated with assertions such as preconditions of the assignments or invariants of the loops.*

$$\frac{\{\{Q\}\}\ S^*\ \{\{R\}\}}{\{\{Q\}\}\ S\ \{\{R\}\}}$$

The next three rules take into consideration the cases when some logical functions of the construction is not well-defined. These rules are the extensions of the well-known versions that use well-defined logical functions. An assertion P will be interpreted as the set of states that satisfy P many times in the following rules. From its context it can be decided which interpretation holds. For example, Q and R are assertions in the expression $Q \implies R$ but they are sets in $Q \subseteq R$.

Theorem 16 *Let* Q *and* R *be two assertions, and* $S_1, \ldots S_n$ *be programs and* $\pi_1, \ldots \pi_n$ *be conditions.*

$$\frac{\begin{array}{c} Q \implies \pi_1 \vee \cdots \vee \pi_n \\ Q \subseteq \mathcal{D}_{\pi_1} \cap \cdots \cap \mathcal{D}_{\pi_n} \\ \forall i \in \{1, \ldots, n\} : \{\{Q \wedge \pi_i\}\}\ S_i\ \{\{R\}\} \end{array}}{\{\{Q\}\}\ \textbf{if}\ \pi_1 \to S_1 \,\square \ldots \square\, \pi_n \to S_n\ \textbf{fi}\ \{\{R\}\}}$$

Proof. We must show that the executions of the conditional statement starting from Q finish at R. Let q be an arbitrary state of Q. Since $Q \subseteq \mathcal{D}_{\pi_1} \cap \cdots \cap \mathcal{D}_{\pi_n}$ and $Q \implies \pi_1 \vee \cdots \vee \pi_n$, according to the definition each execution of the conditional statement starting from q belongs to the executions of the component S_i where its condition π_i is satisfied by q.

These executions terminate in a state satisfying R because $\forall i \in \{1, \ldots, n\} : \{\{Q \wedge \pi_i\}\} S_i \{\{R\}\}$ thus $\{\{Q\}\}\ \textbf{if}\ \pi_1 \to S_1 \,\square \ldots \square\, \pi_n \to S_n\ \textbf{fi}\ \{\{R\}\}$ holds. □

Theorem 17 *Let* Q *and* R *be two assertions, and* S_0 *be a program and* π *be a condition.*

$$\frac{\begin{array}{c} \exists I : A \to \mathbb{L} \text{ and } \exists t : A \to \mathbb{Z} \\ Q \implies I \\ I \subseteq \mathcal{D}_\pi \\ I \wedge \neg\pi \implies R \\ I \wedge \pi \implies t \geq 0 \\ \{\{I \wedge \pi\}\}\ S_0\ \{\{I\}\} \\ \forall c_0 \in \mathbb{Z} : \{\{I \wedge \pi \wedge t = c_0\}\}\ S_0\ \{\{t < c_0\}\} \end{array}}{\{\{Q\}\}\ \textbf{while}\ \pi\ \textbf{do}\ S_0\ \textbf{od}\ \{\{R\}\}}$$

Proof. We need to prove that the executions of the loop starting from Q are finite and they finish at R. Let q be an arbitrary state of Q. Since $Q \implies I$ and $I \subseteq \mathcal{D}_\pi$ thus $q \in \mathcal{D}_\pi$.

The execution starting from q can be splitted into the sections of the executions generated by the body S_0. Each section is started at a state satisfying $I \wedge \pi$ and terminates at a state satisfying I (see the cond. $\{\{I \wedge \pi\}\}S_0\{\{I\}\}$). If the total execution is finite, its last section finishes at a state satisfying $\neg\pi$ or the state q own satisfies $\neg\pi$ if the loop stops at once. It means that each finite execution strating from q finishes at a state of R since $I \wedge \neg\pi \implies R$.

The proof will be complete if we show that there is no infinite execution from q. If there would be an infinite execution, it should consist of infinite sections. Let us consider the infinite sequence of integers that is mapped from the beginning states of the sections by the function t. Because of the criterion $\forall c_0 \in \mathbb{Z} : \{\{I \wedge \pi \wedge t = c_0\}\}\ S_0\ \{\{t < c_0\}\}$ this sequence should be strictly monotone decreasing thus it should contain negative numbers. However all numbers must be nonnegative because of the criterion $I \wedge \pi \implies t \geq 0$. This is a contradiction. □

Theorem 18 *Let* Q *and* R *be two assertions, and* S_0 *be a program and* β *be a condition.*

$$\frac{\begin{array}{c} Q \subseteq \mathcal{D}_\beta \\ \{\{Q \wedge \beta\}\}\ S_0\ \{\{R\}\} \end{array}}{\{\{Q\}\}\ \textbf{await}\ \beta\ \textbf{then}\ S_0\ \textbf{ta}\ \{\{R\}\}}$$

Proof. It is enough to show that the executions of the conditional statement starting from Q finish at R. Let q be an arbitrary state of Q. If $q \in \mathcal{D}_\beta$ and q satisfies β, the executions of the await-statement starting from q are identical to the executions of the atomic region S_0 starting from q. These executions finish at a state in R because of $\{\{Q \wedge \beta\}\}\ S_0\ \{\{R\}\}$. □

The last rule is about the parallel composition.

Theorem 19 *Let* $Q,\ Q_1,\ \ldots\ Q_n$ *and* $R,\ R_1,\ \ldots\ R_n$ *be assertions,* $S_1,\ \ldots\ S_n$ *be programs, and* $S_1^*,\ \ldots\ S_n^*$ *be the annotations of the corresponded programs.*

$$\frac{\begin{array}{c} Q \implies Q_1 \wedge \ldots \wedge Q_n \\ \forall i \in \{1, \ldots, n\} : \{\{Q_i\}\}S_i^*\{\{R_i\}\} \\ \textit{and they are interference free} \\ R_1 \wedge \ldots \wedge R_n \implies R \end{array}}{\{\{Q\}\}\ \textbf{parbegin}\ S_1 \parallel \cdots \parallel S_n\ \textbf{parend}\ \{\{R\}\}}$$

where standard proof outlines $\{\{Q_i\}\}S_i^*\{\{R_i\}\}$, $i \in \{1, \ldots, n\}$, *are called interference free if no normal assignment or atomic region* u *of a component program* S_i *interferes with the proof outline* $\{\{Q_j\}\}S_j^*\{\{R_j\}\}$ *of another component program* S_j *where* $i \neq j$. *We say that* u *does not interfere with* $\{\{Q\}\}S^*\{\{R\}\}$ *if the following conditions are satisfied:*

1. *for all assertions* r *in* $\{\{Q\}\}S^*\{\{R\}\}$ *the formula* $\{\{r \wedge pre(u)\}\}u\{\{r\}\}$ *holds, where* $pre(u)$ *is the precondition of* u *in the annotation* S^*,
2. *for all termination function* $t : \bar{A} \to \mathbb{Z}$ *in* $\{\{Q\}\}S^*\{\{R\}\}$ *where* A *is the base state space of the parallel composition the formula* $\{\{pre(u) \wedge t = c_0\}\}\, u\, \{\{t \leq c_0\}\}$ *holds, where* c_0 *is an arbitrary integer.*

4 Case study

Consider the following problem: given an array x of n integer numbers and an integer number k. Count the elements of x that are divisors of k.
Specification of the problem can be given in the following way:
$A = (x\colon \mathbb{Z}^n, k\colon \mathbb{Z}, count\colon \mathbb{N})$

$Pre = (x = x')$

$$Post = (Pre \wedge count = \sum_{j=1}^{n} \chi(x[j] \mid k))$$

where $\chi\colon \mathbb{L} \to \{0, 1\}$ and $\chi(true) = 1$ and $\chi(false) = 0$

Let S denote the following program:

```
i, count := 1, 0
while i ≤ n do
   if
      x[i] | k → count := count + 1   □
      x[i] ∤ k → skip
   fi;
   i := i+1
od
```

Let Q' denote the intermediate assertion of the sequence S, between the initialisation and the loop, Q'' the intermediate assertion that holds before executing the assignment $i := i + 1$, Inv the invariant and t the variant function of the loop.

Now we shall to prove that $\{\{Pre\}\}\ S\ \{\{Post\}\}$ holds. Since S is a sequence, due to Theorem 14. it is sufficient to prove that

1. $\{\{Pre\}\}\ i, count := 1, 0\ \{\{Q'\}\}$ and
2. $\{\{Q'\}\}$ DO $\{\{Post\}\}$, where DO denotes the loop:
   while $i \leq n$ **do if** $x[i] \mid k \rightarrow count := count + 1 \quad \square\ x[i] \nmid k \rightarrow$ **skip fi**; i:=i+1 **od**
   and $Q' = (Pre \wedge count = 0 \wedge i = 1)$ is given.

Let us prove the two conditions:

1. $\{\{Pre\}\}\ i, count := 1, 0\ \{\{Q'\}\}$
   By replacing i with 1 and $count$ with 0 in Q' we obtain $(Pre \wedge 0 = 0 \wedge 1 = 1)$, that is Pre. Obviously $Pre \implies Pre$ holds. We proved that $Pre \implies Q'^{i \leftarrow 1, count \leftarrow 0}$ holds. Now, by the remark of Theorem 12. $\{\{Pre\}\}\ i, count := 1, 0\ \{\{Q'\}\}$ is deduced.
2. $\{\{Q'\}\}$ DO $\{\{Post\}\}$
   Instead of proving this verification condition, due to Theorem 17. it is sufficient to prove that
   (a) $Q' \implies Inv$ and
   (b) $Inv \subseteq \mathcal{D}_{i \leq n}$ and
   (c) $Inv \wedge \neg(i \leq n) \implies Post$ and
   (d) $Inv \wedge i \leq n \implies n - i \geq 0$ and
   (e) $\forall c_0 \in \mathbb{Z}$: $\{\{Inv \wedge i \leq n \wedge n - i = c_0\}\}\ S_0\ \{\{Inv \wedge n - i < c_0\}\}$

   where S_0 denotes the loop body **if** $x[i] \mid k \rightarrow count := count + 1 \square$ $x[i] \nmid k \rightarrow$ **skip fi**; i:=i+1.
   and the loop invariant Inv and the variant function t are given as follows:
   $Inv = (Pre \wedge i \in [1..n+1] \wedge count = \sum_{j=1}^{i-1} \chi(x[j] \mid k))$
   $t = n - i$
   Let us prove the conditions separately:
   (a) $Q' \implies Inv$
   We have to prove that Q' implies
   i. Pre
   This holds since Q' contains Pre.

ii. $i \in [1..n+1]$
Since $i = 1$, i is an element of the not empty set $[1..n+1]$. The interval $[1..n+1]$ is not empty, because n, that is the length of the array x, is zero or a positive integer number.

iii. $\mathsf{count} = \sum_{j=1}^{i-1} \chi(x[j] \mid k)$
Due to condition Q', $\mathsf{count} = 0$ and $i = 1$. Thus the sum is empty and its value is also 0.

(b) $Inv \subseteq \mathcal{D}_{i \le n}$
Since $i \le n$ is a well-defined logical function, its domain contains all states of the statespace, including those that satisfy Inv.

(c) $Inv \wedge \neg(i \le n) \implies \mathsf{Post}$

i. Pre
The invariant contains the precondition, therefore Inv implies Pre.

ii. $\mathsf{count} = \sum_{j=1}^{n} \chi(x[j] \mid k)$
Since $i \in [1..n+1]$ and $\neg(i \le n)$, therefore we get $i = n+1$.
This, together with $\mathsf{count} = \sum_{j=1}^{i-1} \chi(j \mid k))$ we know from Inv, yields the desired condition.

(d) $Inv \wedge i \le n \implies n - i \ge 0$
This holds because due to the loop condition $i \le n$ is true.

(e) $\forall c_0 \in \mathbb{Z}$: $\{\{Inv \wedge i \le n \wedge n - i = c_0\}\}\ S_0\ \{\{Inv \wedge n - i < c_0\}\}$
Let c_0 be an arbitrary integer number. Since the loop body S_0 is a sequence, we use Theorem 14. with $Inv \wedge i \le n \wedge n - i = c_0$ as Q and with $Inv \wedge n - i < c_0$ as R. It is sufficient to prove the following two conditions:

i. $\{\{Inv \wedge i \le n \wedge n - i = c_0\}\}$ IF $\{\{Q''\}\}$ and
ii. $\{\{Q''\}\}\ i := i + 1\ \{\{P \wedge n - i < c_0\}\}$

where IF denotes the conditional statement **if** $x[i] \mid k \to \mathsf{count} := \mathsf{count} + 1 \square\ x[i] \nmid k \to$ **skip fi**
and the intermediate assertion of the loop body is Q'' is given: $Q'' = Inv^{i \leftarrow i+1} \wedge n - i = c_0$

i. $\{\{Inv \wedge i \leq n \wedge n - i = c_0\}\}$ IF $\{\{Q''\}\}$
Due to Theorem 16., the following conditions are sufficient to prove:
   A. $Inv \wedge i \leq n \wedge n - i = c_0 \implies (x[i] \mid k \vee x[i] \nmid k)$ and
   B. $Inv \wedge i \leq n \wedge n - i = c_0 \subseteq \mathcal{D}_{x[i]|k} \cap \mathcal{D}_{x[i]\nmid k}$ and
   C. $\{\{Inv \wedge i \leq n \wedge n - i = c_0 \wedge x[i] \mid k \wedge n - i = c_0\}\}$ $count := count + 1$ $\{\{Q''\}\}$ and
   D. $\{\{Inv \wedge i \leq n \wedge n - i = c_0 \wedge x[i] \nmid k \wedge n - i = c_0\}\}$ **skip** $\{\{Q''\}\}$

   Let us prove the conditions separately:
   A. $Inv \wedge i \leq n \wedge n - i = c_0 \implies (x[i] \mid k \vee x[i] \nmid k)$
   For each state of the statespace for which $Inv \wedge i \leq n \wedge n - i = c_0$ holds, either $x[i]$ is a divisor of k or $x[i]$ is not a divisor of k.
   B. $Inv \wedge i \leq n \wedge n - i = c_0 \subseteq \mathcal{D}_{x[i]|k} \cap \mathcal{D}_{x[i]\nmid k}$
   In order to ensure that $x[i] \mid k$ and $x[i] \nmid k$ are well-defined functions, we have to take into account not only that the divisibility $x[i] \mid k$ can be answered only if $x[i]$ is not zero, but the index i has to be inside the bounds of the array x. More precisely, we want to prove that
   $Inv \wedge i \leq n \wedge n - i = c_0 \implies i \in [1..n] \wedge x[i] \neq 0 \wedge i \in [1..n] \wedge x[i] \neq 0$.
   Although $i \in [1..n+1]$ (due to the invariant) and the loop condition $i \leq n$ together allow us to deduce that $i \in [1..n]$ holds, we cannot guarantee that each state of the statespace for which $Inv \wedge i \leq n \wedge n - i = c_0$ holds, $x[i]$ is not 0. The reason is, that there is no assumption for the elements of x, except that they are integer numbers. The case when $x[i] = 0$, is not excluded by any condition we know and are allowed to use. If $x[i]$ equals 0, the expressions $x[i] \mid k$ and $x[i] \nmid k$ have no defined value. The current condition cannot be proven. We provide the remaining part of the proof for the sake of completeness.
   C. $\{\{Inv \wedge i \leq n \wedge n - i = c_0 \wedge x[i] \mid k \wedge n - i = c_0\}\}$ $count := count + 1$ $\{\{Q''\}\}$
   Let us recall that Q'' is $(Inv^{i \leftarrow i+1} \wedge n - i = c_0)$.
   $Q''^{count \leftarrow count+1} = (Pre \wedge i + 1 \in [1..n+1] \wedge count + 1 = \sum_{j=1}^{i-1} \chi(x[j] \mid k)) + \chi(x[i] \mid k))$. By Theorem 12. it is sufficient

to prove that
$(Inv \wedge i \leq n \wedge n - i = c_0 \wedge x[i] \mid k \wedge n - i = c_0) \implies Q''^{count \leftarrow count+1}_{count}$.

- Pre
  Pre in included in Inv.
- $i + 1 \in [1..n + 1]$
  Due to the invariant $i \in [1..n + 1]$ holds. This, combined with the loop condition $i \leq n$ implies $i + 1 \in [1..n + 1]$. $i = 1$ and $i \leq n$.
- $count + 1 = \sum_{j=1}^{i-1} \chi(x[j] \mid k)) + \chi(x[i] \mid k))$
  By the loop invariant Inv, $count = \sum_{j=1}^{i-1} \chi(x[j] \mid k))$. Since in this case $x[i]$ is a divisor of k, $\chi(x[i] \mid k) = 1$, we added 1 to both sides of the previous equation.

D. $\{\{Inv \wedge i \leq n \wedge n - i = c_0 \wedge x[i] \nmid k \wedge n - i = c_0\}\}$ **skip** $\{\{Q''\}\}$
Let us recall that Q'' is $(Inv^{i \leftarrow i+1} \wedge n - i = c_0)$ that is $(Pre \wedge i + 1 \in [1..n + 1] \wedge count = \sum_{j=1}^{i-1} \chi(x[j] \mid k)) + \chi(x[i] \mid k) \wedge n - i = c_0)$. By Theorem 11. it is sufficient to prove that
$(Inv \wedge i \leq n \wedge n - i = c_0 \wedge x[i] \nmid k \wedge n - i = c_0) \implies Q''$.

- Pre
  Pre in included in Inv.
- $i + 1 \in [1..n + 1]$
  We prove this in the same way as we did in the previous case.
- $count = \sum_{j=1}^{i-1} \chi(x[j] \mid k)) + \chi(x[i] \mid k))$
  By the loop invariant Inv, $count = \sum_{j=1}^{i-1} \chi(x[j] \mid k))$. Since in this case $x[j]$ is not a divisor of k, $\chi(x[i] \mid k)$ evaluates to zero. The desired condition holds because both sides of the equation $count = \sum_{j=1}^{i-1} \chi(x[j] \mid k))$ in Inv remained the same.
- $n - i = c_0$

It is obviously true, since it is on the left side.

ii. $\{\{Q''\}\}\ i := i + 1\ \{\{Inv \wedge n - i < c_0\}\}$
$Q'' = (Inv^{i \leftarrow i+1} \wedge n - i = c_0$. It is obvious that $(Inv^{i \leftarrow i+1} \wedge n - i = c_0) \implies (Inv \wedge n - i < c_0)^{i \leftarrow i+1}$, therefore by Theorem 12 we get the expected correctness formula.

To prove the correctness formula $\{\{Q\}\}\ S\ \{\{R\}\}$, all the verification conditions generated by the verification rules have to be satisfied. Let us remember that the following condition was not proven:
$Inv \wedge i \leq n \wedge n - i = c_0 \subseteq \mathcal{D}_{x[i]|k} \cap \mathcal{D}_{x[i]\nmid k}$
We could not guarantee that both of the logical functions $x[i] \mid k$ and $x[i] \nmid k$ are well-defined functions. Evaluating these functions of the program might lead to abortion. The rest of the conditions were unnecessary to prove, their proof was given for the sake of completeness.

5 Summarization

The main idea behind this work is to take into account abortion caused by partial functions in programs and extend verification rules to be able to ensure that such programs are total correct. To reason about correctness, we provided the formal definition of the semantics of programs under our investigation. One of the contributions of this paper is, that the semantics of program constructs are defined in such a way that they suit an existing relational model of programming. This relational model defines the program as a set of its possible executions and also provides definition for other important programming notions like problem and solution. Then, we provide a verification rule for each class of statements. The first six rules are well-known. Three rules are new, they are presented along with their proofs. The use of the rules is demonstrated by means of a verification case study.

References

[1] K. R. Apt, E.-R. Olderog, *Verification of Sequential and Concurrent Program*, Springer-Verlag, 1997. ⇒67

[2] E. W. Dijkstra, *A Discipline of Programming*, Prentice-Hall, Englewood Cliffs, New York, 1976. ⇒65, 68, 73

[3] Á. Fóthi, Mathematical Approach to Programming, *Ann. Univ. Sci. Budapest. Sect. Comput.* **9** (1988) 105–114. ⇒65, 68

[4] Á. Fóthi et al, Some concepts of a Relational Model of Programming, *Proc. 4th Symposium on Programming Language and Software Tools*, Visegrád, Hungary, June 8-14, 1995. (ed. Varga L.,) pp. 434–446, ⇒65
[5] Á. Fóthi, *Bevezetés a programozáshoz*, ELTE Eötvös Kiadó. 2005. (in Hungarian). ⇒65, 68
[6] T. Gregorics, Concept of abstract program, *Acta Universitatis Sapientiae, Informatica*, **4,** 1 (2012) 7–16. ⇒68, 69
[7] D. Gries, S. Owicki, An axiomatic proof technique for parallel programs, *Acta Inf.*, **6,** 4 (1976) 319–340. ⇒65, 73
[8] D. Gries, *The Science of Programming*, Springer, Berlin, 1981. ⇒65, 67, 73
[9] C. A. Hoare, An axiomatic basis for computer programming, Comm. of the ACM **12,** 10 (1969) 576–580. ⇒65, 67, 73
[10] Z. Manna, *Mathematical theory of computation*, McGraw Hill, 1974. ⇒67
[11] W.-P. de Roever et al, *Concurrency Verification*, Cambridge University Press, 2001. ⇒67

Received: June 12, 2017 • Revised: July 8, 2017

DE GRUYTER OPEN
Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 83–96

DOI: 10.1515/ausi-2017-0006

Analysis of Sci-Hub downloads of computer science papers

Darko ANDROČEC
Faculty of Organization and Informatics, University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
email: dandrocec@foi.hr

Abstract. The scientific knowledge is disseminated by research papers. Most of the research literature is copyrighted by publishers and available only through paywalls. Recently, some websites offer most of the recent content for free. One of them is the controversial website Sci-Hub that enables access to more than 47 million pirated research papers. In April 2016, Science Magazine published an article on Sci-Hub activity over the period of six months and publicly released the Sci-Hub's server log data. The mentioned paper aggregates the view that relies on all downloads and for all fields of study, but these findings might be hiding interesting patterns within computer science. The mentioned Sci-Hub log data was used in this paper to analyse downloads of computer science papers based on DBLP's list of computer science publications. The top downloads of computer science papers were analysed, together with the geographical location of Sci-Hub users, the most downloaded publishers, types of papers downloaded, and downloads of computer science papers per publication year. The results of this research can be used to improve legal access to the most relevant scientific repositories or journals for the computer science field.

Computing Classification System 1998: K.7.4, K.3.2
Mathematics Subject Classification 2010: 62P30
Key words and phrases: computer science ethics, journal paywalls, pirated papers, scientific publishing

1 Introduction

Access to the research literature is essential to the successful work of researchers and the education of the general public [1]. Scientists and the general public rely on a wide range of channels to access the research literature [2]: printed issues of journals, interlibrary loans, publishers' online platforms such as Elsevier ScienceDirect, preprint repositories such as ArXiv, institutional and authors' web pages. Many now propose free access to all scientific papers to everybody, including the European Union where consensus of all members' ministers of science, innovation, trade and industry was made in May 2016 that all scientific papers founded by the EU should be freely available by 2020 [3]. People failing to retrieve articles through these channels use article requests from authors by e-mail or using social media such as Academia.edu, Mendeley and ResearchGate. Recently, some online repositories offer most of the scientific papers for free. One of them is Sci-Hub. Sci-Hub website currently hosts more than 47 million pirated research papers and has millions of visitors per month. Online piracy represents a copyright infringement whereby copyright material (scientific articles in this case) is reproduced or distributed without appropriate permission [4]. The Sci-Hub founder Alexandra Elbakyan from Kazakhstan claims that the main aim of the Sci-Hub project is to circumvent the copyright restrictions to speed up the development of science, especially in developing countries where researchers do not have institutional access to publishers' paywalls. Of course, publishers have a different opinion, e.g. Elsevier sued Sci-Hub and its founder, and frequently requested its web domains to be put down. Publishers claim that journals have costs, even if they do not pay researchers - authors and reviewers, e.g. editors (and sometimes copy editors, proofreaders, illustrators) are mostly paid professionals, digital publishing is nowadays expensive, and article usage information is lost when using Sci-Hub and/or similar websites [5]. Sci-Hub website is controversial, but many researchers from all parts of the world (even from countries and institutions that have legal access to research papers) are using its services [6]. "Over the 6 months leading up to March, Sci-Hub served up 28 million documents" [6]. The six-month log data (September 2015-February 2016) from Sci-Hub website are now publicly available at [7].

Bohannon [6] presented an aggregate view that relies on all downloads and for all fields of study, but his findings might be hiding interesting patterns within computer science, and this is still gap in the current literature. In this paper, the mentioned data was analysed in more detail and with computer science papers in the focus. The main research questions are: Who are Sci-

Hub users that download computer science research papers and where are they from? What computer science research papers did the mentioned users download? The results of this research can be used to identify the most relevant scientific repositories, journals, and conference proceedings for the computer science field, and which of the sources is more inaccessible to researchers.

This paper proceeds as follows. First, in Section 2, the related work is listed. The next section presents the used research methodology. The results are shown in Section 4. The conclusions are provided in the final section.

2 Related work

Bohannon [6] presented the statistics of the Sci-Hub usage based on extensive server log data [7] supplied by the Sci-Hub creator, Alexandra Elbakyan. To protect the privacy of Sci-Hub users, their geographical locations were aggregated to the nearest town using Google Maps, so IP addresses are not contained in the publicly released data set [7]. Bohannon's first conclusion is that Sci-Hub users are not limited to the developing world, e.g. a quarter of the Sci-Hub requests came from OECD members that should have the best journal access.

Parkhill [8] loaded the top 100 downloads from the Sci-Hub data set [7] into tool PlumX. Most of the downloaded papers are from 2015, so Sci-Hub users were downloading the most recently published papers. Physical sciences and engineering, together with life sciences, garnered most of the downloads. Babutsidze [9] examined the data from illegal downloads of economic content from Sci-Hub, based on the log data from [7]. He concentrated on downloads of the top five economics journals: American Economic Review, Quarterly Journal of Economics, Journal of Political Economy, Econometrica and Review of Economic Studies. Babutsidze [9] concluded that there is a very small number of downloads of economics papers from Sci-Hub, most of the downloads are from under-developed countries, and open access economics papers were downloaded from Sci-Hub because of convenience.

Cabanac [1] reveals that 36% of all digital object identifier (DOI) articles are available for free at Library Genesis platform (LibGen). For the three major publishers (Elsevier, Springer and Wiley) the percentage is even higher (68%). As of January 2014, LibGen hosted and distributed 25 million digital documents, mainly for educational purposes. Users crowdsource articles to LibGen directly or indirectly through services such as Reddit Scholar and Sci-Hub. Cabanac [1] also claims that people crowdsource articles through various

other channels such as Reddit, Sci-Hub, #icanhazpdf hashtag on Twitter, personal websites and text-sharing platforms.

Swab and Romme [10] analysed the use of #icanhazpdf hashtag as a means of obtaining health science literature. They have used RowFeeder software to monitor #icanhazpdf requests between 1 February and 30 April 2015, and they concluded that there were 302 requests for health sciences literature in this period. This number accounts for a relatively small proportion of paper sharing compared with other online platforms.

Gardner and Gardner [11] surveyed users of Twitter, Reddit Scholar and Facebook about crowdsourced research, their demographic information, frequency of use and motivations. They concluded that primary platforms used to organize crowdsourcing of research articles are Twitter, Facebook and Reddit, and the primary websites to host the content were AvaxHome, LibGen and Sci-Hub. Elsevier, Springer and Wiley account for 83% of all LibGen's content. The majority of respondents claim to obtain articles for utilitarian reasons, and crowdsourcing is their preferred alternative to using interlibrary loans.

Timus and Bautsidze [4] examined the Sci-Hub downloads data to uncover patterns in piracy in the European Studies research. They relied on the information provided by the University Association for Contemporary European Studies (UACES) that provides the list of European Studies journals. For their analysis, they have chosen the journals with ISI impact factor greater than 1. Their analysis reveals that the readers are mostly interested in subjects reflecting the current European challenges such as populism, extremism and the economic crisis.

3 Research methodology

The publicly available Sci-Hub's server log data available at [7] was downloaded and imported into MySQL database system. The mentioned data set was already anonymised (there are no IP addresses, IP addresses were aggregated to the nearest city location). For the efficiency reasons, the data was put into six tables (scihub_data1 - scihub_data6), one table for each monthly server log data. The data consists of date and time, digital object identifier (DOI), country, town and geographical position. DOIs are not tagged by subject or keyword so it is not possible to return a set of DOIs for the computer science field. Therefore, identifying the articles from computer science field represents a challenge. For this reason, all the DOIs from DBLP were extracted in XML

format on May 23, 2016, and imported into another MySQL table. DBLP service provides open bibliographic information on major computer science journals and proceedings. "The DBLP Computer Science Bibliography of the University of Trier has grown from a very specialized small collection of bibliographic information to a major part of the infrastructure used by thousands of computer scientists" [12]. As of May 2016, DBLP indexes over 3.3 million publications, published by more than 1.7 million authors. To this end, DBLP indexes about 32,000 journal volumes, more than 31,000 conference or workshop proceedings and more than 23,000 monographs.

The DBLP data was downloaded in the XML format. The Java classes were developed to parse the DBLP's XML file and extract the DOIs data into CSV file that was used to import the DBLP's DOIs data into MySQL database. The list of all DOIs from DBLP computer science bibliography was saved into a new table (dblp_dois), and it is assumed that all the main computer science works are included. Of course, some works may not be in this catalogue, which is the limitation of this study, but the situation in other science fields is even worse, e.g. comprehensive meta-indices are missing for most areas of science [12]. The DOIs fields in both tables were defined as database indexes, to enable better query performance. Next, the SQL queries were used to find an intersection of two data sets (Sci-Hub log data and DBLP's DOIs).

First, the views containing the intersection of each of the six scihub_data tables with dblp_dois were created.These tables were very big, and it was impractical (time-consuming) to analyse the data, because SQL queries on the views in some cases run for several hours. Table 1 shows the number of rows in each of the six tables containing Sci-Hub data, and the number of rows of views showing the intersection with DBLP data. Only 5.95% of the data downloaded from Sci-Hub were computer science papers. For this reason, separate tables for each of the six views were created.

The creation of each table took several hours, but after that the complete intersection data was in tables, indexes were put on relevant fields (DOI and country), and SQL queries needed to analyse the results took reasonable execution time (a few minutes). This data (Sci-Hub downloads of computer science papers) is publicly available at https://github.com/dandrocec/in the SciHub-ComputerScience repository in the form of SQL scripts.

Table	Total	in DBLP	%
scihub_data1	3 759 219	217 689	5.79
scihub_data2	6,017,112	407,387	6.77
scihub_data3	4,774,085	279,952	5.86
scihub_data4	1,837,701	97,042	5.28
scihub_data5	5,876,395	325,795	5.54
scihub_data6	4,752,852	280,770	5.91
Sum	27,017,364	1,608,635	5.95

Table 1: Intersection of Sci-Hub and DBLP data

4 Results

The analysis was done on the six tables (data1-data6) that represent the intersection of Sci-Hub log data and DBLP DOIs data, i.e. the computer science papers downloaded at Sci-Hub web page in a six-month time frame. One of the research questions of this paper is: What computer science research papers did the mentioned users download? First, the ranking of the most downloaded computer science papers from Sci-Hub had to be obtained. For the purpose of performing queries on all data, one view with all the data was created Then, the most downloaded computer science papers were obtained by using the SQL query.

In total, 607,023 computer science papers were downloaded from Sci-Hub website in a six-month period. Table 2 shows twenty most downloaded computer science papers from Sci-Hub. Fourteen of the mentioned papers are journal papers, three are conference papers, two are chapters in scientific books, and one is a technical report. Eleven papers are published in IEEE's publications, three in Springer's publications, two in Elsevier's, and one paper in MIT's, Wiley's, SIAM's and ArXiv's publications, respectively. Most of the downloaded papers are new, the only exception being "How to Construct Pseudorandom Permutations from Pseudorandom Functions" from 1988. The most downloaded papers are aligned with currently popular themes in computer science, e.g. titles of five papers contain the phrase "Internet of things". Some of the papers are from open-access journals, but readers still decide to use Sci-Hub instead of journals' official web pages, so paid access is not the only problem. The reason might be that Sci-Hub users do not bother with which of the articles is open-access, they just use Sci-Hub website to retrieve all the necessary articles from one place.

Next, the authors of the top 20 most downloaded papers were extracted to analyse how many total downloaded articles these authors have. Data contains

DOI	Title	Downloads
10.1109/ISPASS .2015.7095803	Nyami: a synthesizable GPU architectural model for general-purpose and graphics-specific workloads	1,118
10.1007/s11948-014-9521-4	Penetrating the Omerta of Predatory Publishing: The Romanian Connection	746
10.1109/TKDE .2013.109	Data mining with big data	725
10.1007/978-3-319-28658-7_55	Detection of Copy-Move Forgery in Images Using Segmentation and SURF	656
10.1162/jocn_a _00880	The Role of Dopamine in Temporal Uncertainty	457
10.1109/ACCESS. 2014.2362522	Information Security in Big Data: Privacy and Data Mining	451
10.1016/j.comnet .2010.05.010	The Internet of Things: A survey	343
10.1109/CTS .2014.6867550	Defining architecture components of the Big Data Ecosystem	307
10.1109/TMI .2013.2265603	Deformable medical image registration: a survey	307
10.1016/j.neunet .2014.09.003	Deep Learning in Neural Networks: An Overview	305
10.1137/0217022	How to Construct Pseudorandom Permutations from Pseudorandom Functions	301
10.1109/JIOT .2014.2306328	Internet of Things for Smart Cities	291
10.1007/978-3-642-55032-4_6	Do Personality Traits Work as Moderator on the Intention to Purchase Mobile Applications Work? - A Pilot Study	288
10.1109/COMST .2015.2444095	Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications	281
10.1109/MobileCloud .2015.40	Cloud Computing for Emerging Mobile Cloud Apps	281
10.1002/asi.23445	Bibliogifts in LibGen? A study of a text-sharing platform driven by biblioleaks and crowdsourcing	271
10.1016/j.future .2013.01.010	Internet of Things (IoT): A vision, architectural elements, and future directions	259
10.1109/TIE .2006.878356	Power-Electronic Systems for the Grid Integration of Renewable Energy Sources: A Survey	227
10.1109/JPROC .2014.2371999	Software-Defined Networking: A Comprehensive Survey	224
10.1109/JIOT .2014.2312291	Research Directions for the Internet of Things	214

Table 2: Twenty most downloaded computer science papers

only DOIs, so the Java program was created to parse the DBLP text file to extract all DOIs of the mentioned authors and to create SQL query to check how many times these articles where downloaded from SciHub during time period of publicly available SciHub's log data. The analysis have shown that

other articles of the same authors were not downloaded in great numbers. So, the SciHub users mostly search for a particular article, not all papers of the particular author. The most downloaded are articles from the following authors: Xindong Wu (1280 downloads), Timothy N. Miller (1124), Jeff Bush (1118), Philip Dexter (1118), Aaron Carpenter (1118), Xingquan Zhu (944), Wei Ding (865), Rajkumar Buyya (795), Gong-Qing Wu (776), Dragan Djuric (760), V. T. Manu (672), B. M. Mehtre (656), Lei Xu (519), Jian Wang (513), etc. Most of the authors have small number of papers, and one paper (the one from the list of the 20 most downloaded articles) has more the 90% of authors' total SciHub's downloads. If we look at Google Scholar's citations (on 14th February 2017) of the mentioned authors there is also no correlation: some authors are often cited (e.g. Rajkumar Buyya - 56076 citations and Xindong Wu - 15611 citations), and some have low number of citations (e.g. V. T. Manu - 11 citations). Of course, some authors are junior researchers with small number of recent publications, and the high number of citations of their works cannot be expected yet.

Next analysis, the list of twenty top countries per Sci-Hub downloads of computer science papers is presented in Table 3. The top five countries with the most downloads were India, Iran, China, the United States and Indonesia. Most of the countries on this list are developing countries, but the OECD countries are also there.

To get more insight, the countries' download data was normalized by downloads per 100,000 population. The population was obtained from Wikipedia data (last estimated value) on 9 September 2016. The results are shown in Table 4. In this case, the top five countries include Tunisia, Iran, Greece, Morocco and Jordan. The only European country in top five is Greece, a country that has been in serious financial crisis for many years now. The reason why Sci-Hub is so popular in Greece could be that the access to scientific database is worse than it was before the country started to experience serious financial crisis. Also, some central European countries such as Serbia, Croatia and Slovenia are high on the list. Mapping IP addresses to real-world locations can paint a false picture if people hide behind web proxies or anonymous routing services. But according to Elbakyan, fewer than 3% of Sci-Hub users are using those [6].

Next, the detailed information of computer science papers was retrieved by their DOIs. For this purpose, a Java class was developed which invokes CrossRef DOI REST application programming interface. The CrossRef REST call returns JSON information about a specific resource (paper identified by its DOI), e.g. for the identified most downloaded paper (Nyami: a synthesizable

Country	Downloads
India	424,652
Iran	231,691
China	94,422
The United States	67,336
Indonesia	56,751
Egypt	52,264
Morocco	51,326
Russia	46,659
Pakistan	45,859
Germany	43,332
Tunisia	33,917
Vietnam	26,913
Brazil	26,255
France	25,640
Malaysia	21,596
Greece	19,783
Algeria	17,405
Canada	13,677
Jordan	11,553
The Netherlands	10,513

Table 3: Top downloads per country

GPU architectural model for general-purpose and graphics-specific workloads). In total, 607,023 computer science papers were downloaded from Sci-Hub website in a six-month period, so the invocation of the CrossRef REST web service took some time. The list of DOIs was split into six partitions, and the Java class was executed during several nights to retrieve and parse all the data. The JSON response was parsed with the aim to receive publication date, publisher and paper type information. The results were stored in CSV files and imported into single Microsoft Excel file for further analysis. First, the analysis by publishers of computer science papers downloaded at Sci-Hub site was performed. The results are shown in Figure 1. The most downloaded publishers were Institute of Electrical and Electronics Engineers (IEEE) with 169,313 papers, Elsevier BV with 132,098, and Springer Nature with 109,586 papers. These results can be compared with Bohannon's general (all scientific fields) analysis [6] where he concluded that three most downloaded publishers were Elsevier, Springer and IEEE. The fact that IEEE is the most downloaded publisher in computer science field is expected, because the computer science and electrical engineering is its main focus.

The next analysis involves downloaded papers by years. The results in Figure

Country	Downloads	Population	Normalized
Tunisia	33,917	10,982,754	308.8205381
Iran	231,691	79,200,000	292.5391414
Greece	19,783	10,955,000	180.5842081
Morocco	51,326	33,848,242	151.6356448
Jordan	11,553	9,710,752	118.9712187
Switzerland	7,968	8,341,000	95.52811413
Portugal	9,136	10,341,330	88.34453595
Serbia	4,889	7,041,599	69.43025299
Malaysia	21,596	31,192,000	69.23570146
Croatia	2,672	4,190,700	63.76023099
Singapore	3,513	5,535,000	63.46883469
The Netherlands	10,513	17,000,059	61.84096185
Lithuania	1,765	2,866,935	61.56400476
Latvia	1,177	1,973,700	59.63418959
Ireland	3,676	6,378,000	57.63562245
Egypt	52,264	91,688,000	57.00200681
Hong Kong	3,907	7,234,800	54.00287499
Germany	43,332	82,175,700	52.7309168
Slovenia	1,022	2,063,077	49.53765662
Algeria	17,405	40,400,000	43.08168317

Table 4: Countries' downloads normalized per population

2 show that the most recently published papers were the most popular to download from Sci-Hub site. Most papers were downloaded in 2015, which is not surprising since the available Sci-Hub dataset spans downloads from September 2015 till February 2016.

Finally, the types of downloaded papers were analysed (Figure 3). Journal papers were dominant (58% of computer science papers downloaded from Sci-Hub website were journal papers). The reason for this can be seen in limited accessibility of researchers to chosen journals and perceived quality of papers published in established journals. The next types of papers are conference proceeding articles (31%) and book chapters (10%).

5 Conclusion

Many scientists, especially from developing countries, cannot access the research papers they need. Researchers who fail to retrieve articles through publisher's paywalls, use other, possibly illegal methods to get the needed materials. One of them is Sci-Hub website, hosting more than 47 million pirated research papers. In this work, publicly available data of a six-month

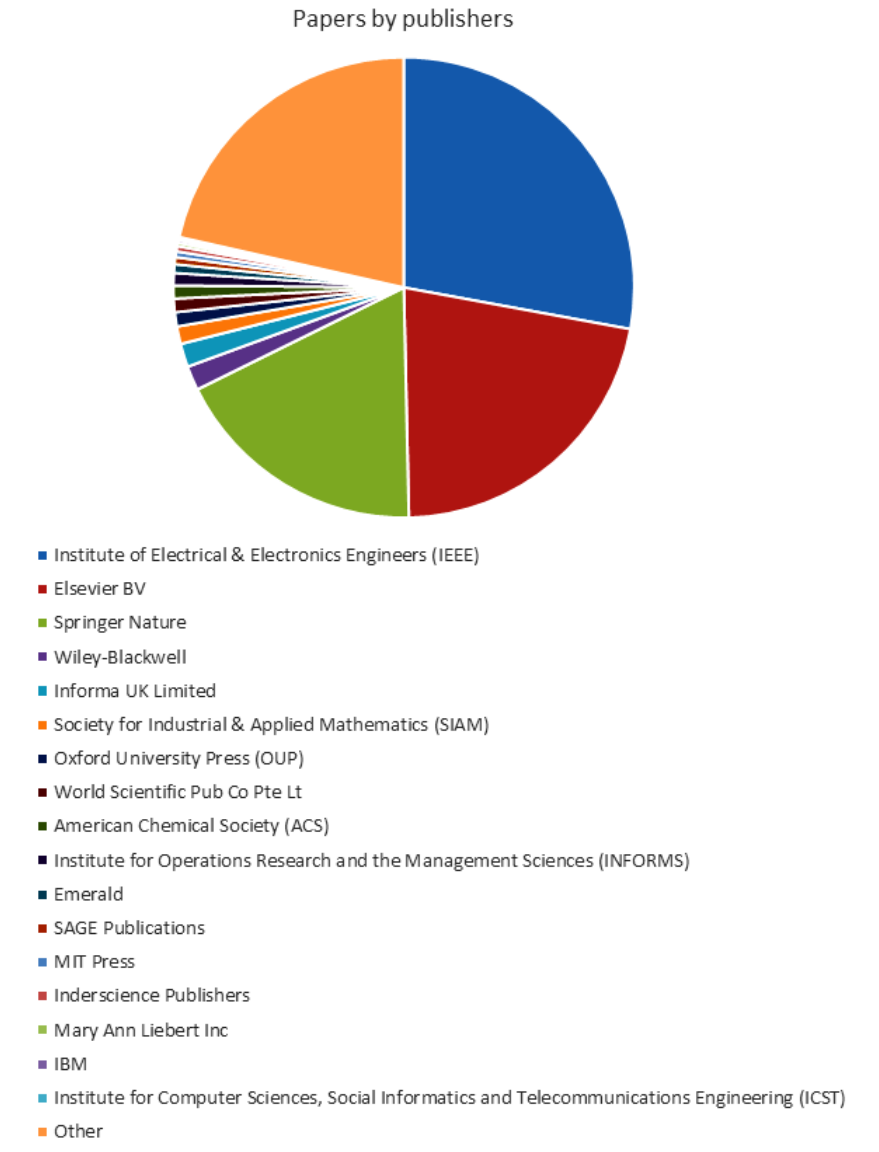

Figure 1: Downloaded papers by publishers

server log data was used to analyse the most downloaded computer science papers in this period with the aim to identify the most important repositories and journals for computer science community. To achieve this task, the

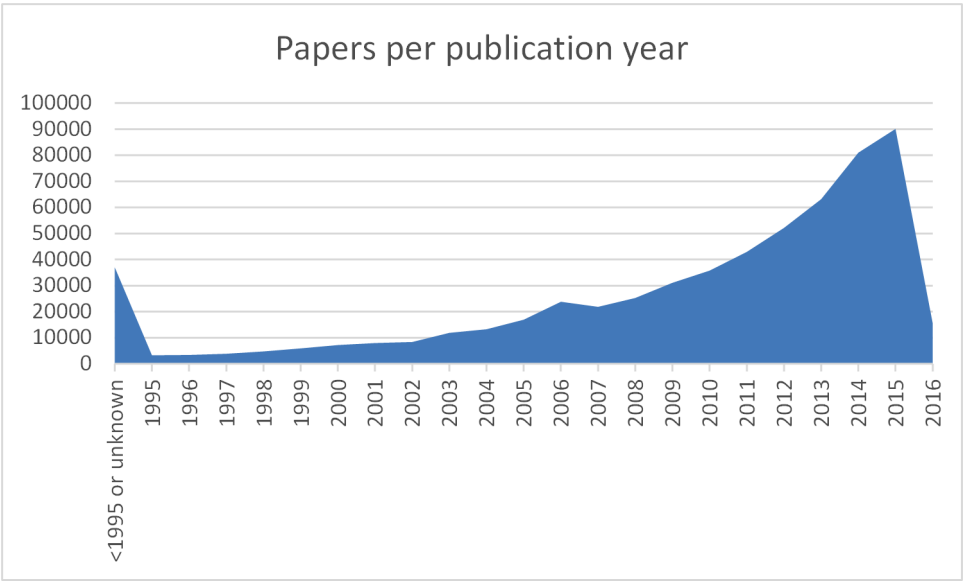

Figure 2: Downloaded computer science papers per publication year

DOIs from the available Sci-Hub log server data were intersected with the DOIs from DBLP (computer science bibliography of the University of Trier), to analyse only log entries of computer science papers. It was assumed that all main computer science works are included in DBLP. The limitation of this study is that some works may not be in DBLP catalogue, but the situation is worse in other science fields because there are no meta-indices for most areas of science. In total, 607,023 computer science papers were downloaded from Sci-Hub in the six-month period.

Table 2 of this work shows the twenty most downloaded computer science papers from Sci-Hub. Next, the top five countries with the most downloads were India, Iran, China, the United States and Indonesia. If the data is normalized by downloads per population, then the ranking of the countries is different and top five countries include Tunisia, Iran, Greece, Morocco and Jordan. Next, the detailed information of computer science papers was retrieved by their DOIs and CrossRef DOI API. The most downloaded publishers were Institute of Electrical and Electronics Engineers (IEEE) with 169,313 papers, Elsevier BV with 132,098 and Springer Nature with 109,586 papers. If the countries, institutions or libraries tried to improve computer scientists' work conditions, they should consider enabling access to repositories or journals of the men-

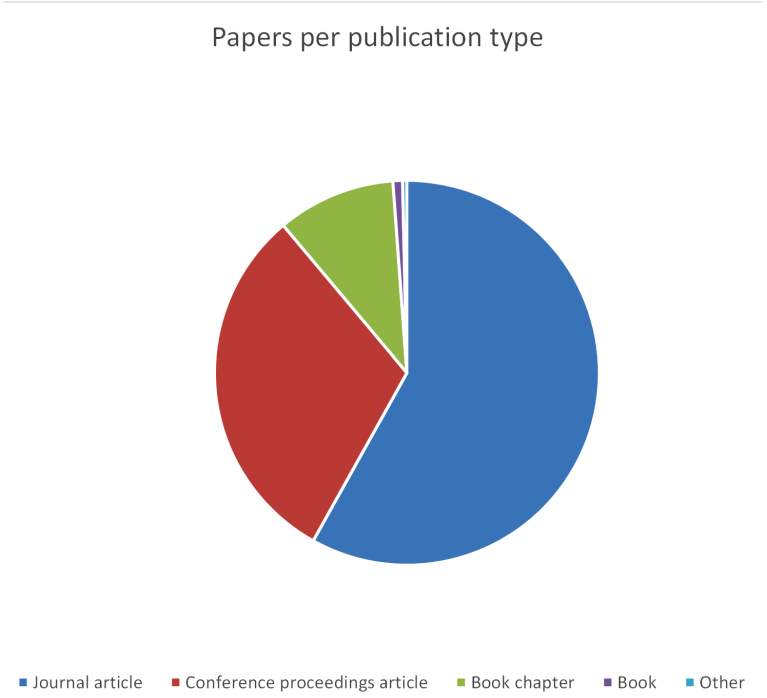

Figure 3: Types of downloaded papers

tioned publishers first. Furthermore, the most recently published computer science papers were the most popular to be downloaded from Sci-Hub site. If the types of computer science papers downloaded from Sci-Hub were observed, the journal papers were dominant (58%), followed by conference proceeding articles (31%) and book chapters (10%). The conclusion can be made that access to quality journal papers is the most important for computer scientists. The findings of this paper show who Sci-Hub users that download computer science research papers are and where they are from; and what computer science research papers the mentioned users downloaded. This information is useful for libraries and policies that make decisions on the most important sources (online repositories and journals) for computer science community. It is also important for publishers who can consider new methods how to make access to the most wanted content more accessible and cheaper for their end-users.

References

[1] G. Cabanac, Bibliogifts in LibGen? Study of a text-sharing platform driven by biblioleaks and crowdsourcing, *Journal of the Association for Information Science and Technology* **67,** 4 (2016) 874–884. ⇒84, 85

[2] P. M. Davis, W. H. Walters, The impact of free access to the scientific literature: a review of recent research, *Journal of the Medical Library Association* 99, 3(2011) 208–217. ⇒84

[3] M. Enserink, In dramatic statement, European leaders call for 'immediate' open access to all scientific papers by 2020, *Science,* May. 27, 2016. ⇒84

[4] N. Timus, Z. Babutsidze, Pirating European Studies, *Journal of Contemporary European Research* **12,** 3 (2016) 783–791. ⇒84, 86

[5] M. McNutt, My love-hate of Sci-Hub, *Science,* **352,** 6285 (2016) 497–497. ⇒84

[6] J. Bohannon, Who's downloading pirated papers? Everyone, *Science.* April 28, 2016 . ⇒84, 85, 90, 91

[7] A. Elbakyan, J. Bohannon, Data from: Who uses Sci-Hub? Everyone, *Dryad Digital Repository.* April 28, 2016. ⇒84, 85, 86

[8] M. Parkhill, Sci-Hub: The academic cat is out of the bag, *Plam Analytics*, May 16, 2016 ⇒85

[9] Z. Babutsidze, Pirated economics, *MPRA paper* 71703. 2016. ⇒85

[10] M. Swab, K. Romme, Scholarly sharing via Twitter: #icanhazpdf requests for health sciences Literature, *Journal of the Canadian Health Libraries Association/Journal de l'Association des bibliotheques de la sante du Canada* **37,** 1 (2016) 6–11. ⇒86

[11] C. C. Gardner, G. J. Gardner, Fast and furious (at publishers): the motivations behing crowdsourced research sharing, *College & Research Library* January 2017, pp. 1–24. ⇒86

[12] M. Ley, The DBLP computer science bibliography: evolution, research issues, perspectives, in *String Processing and Information Retrieval*, LNCS 2476 (2002) 1–10 (eds. A. H. F. Laender, A. L. Oliveira), Springer Berlin Heidelberg. ⇒87

Received: January 14, 2017 • Revised: February 14, 2017

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University
De Gruyter Open
Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/).

References should be listed alphabetically based on the Intructions for Authors given at the address http://www.acta.sapientia.ro.

Illustrations should be given in Encapsulated Postscript (eps) format.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Idea Printing House
Director: Péter Nagy

ISSN 1844-6086
http://www.acta.sapientia.ro