BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 19 2013, pages 2490–2493
doi:10.1093/bioinformatics/btt407

Sequence analysis

Advance Access publication July 12, 2013

RACER: Rapid and accurate correction of errors in reads

Lucian Ilie* and Michael Molnar

Department of Computer Science, University of Western Ontario, N6A 5B7 London, ON, Canada

Associate Editor: Gunnar Ratsch

ABSTRACT

Motivation: High-throughput next-generation sequencing technologies enable increasingly fast and affordable sequencing of genomes and transcriptomes, with a broad range of applications. The quality of the sequencing data is crucial for all applications. A significant portion of the data produced contains errors, and ever more efficient error correction programs are needed.

Results: We propose RACER (Rapid and Accurate Correction of Errors in Reads), a new software program for correcting errors in sequencing data. RACER has better error-correcting performance than existing programs, is faster and requires less memory. To support our claims, we performed extensive comparison with the existing leading programs on a variety of real datasets.

Availability: RACER is freely available for non-commercial use at www.csd.uwo.ca/~ilie/RACER/.

Contact: ilie@csd.uwo.ca

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on February 20, 2013; revised on April 19, 2013; accepted on July 8, 2013

1 INTRODUCTION

The automated Sanger sequencing method (Sanger *et al.*, 1977) has revolutionized biological research by unveiling the sequence of the DNA molecule, most prominently that of the human genome. Limitations of the method created the need for improved sequencing technologies. Great demand caused the discovery of several so-called next-generation sequencing (NGS) technologies, such as Illumina/Solexa, Roche/454, Life/APG's SOLiD, Helicos BioSciences' HeliScope, Pacific Biosciences and Life's Ion Torrent; the survey of Metzker (2010) gives detailed descriptions.

These high-throughput technologies produce huge amounts of data at decreasing costs, thus enabling an ever increasing number of applications, including *de novo* genome assembly, genome resequencing, cancer mutation discovery, metagenomics, DNA–protein interaction discovery and so forth. The way has been opened for ambitious projects, such as the 1000 Genomes Project (Siva, 2008), the first project to sequence the genomes of a large number of people to provide a comprehensive resource on human genetic variation, and the Genome 10K Project (Haussler *et al.*, 2009), aiming at discovering the genomes of 10 000 vertebrate species.

A large number of bioinformatics programs are essential in analyzing the NGS data. Although their variety is impressive, they all need high-quality data. Nearly half of the reads produced by Illumina, the currently dominant technology, contain errors. Among other problems, they make genome assembly much more difficult, and genome assemblers, such as Euler-SR (Chaisson and Pevzner, 2008), ALLPATHS (Butler *et al.*, 2008), ABySS (Simpson *et al.*, 2009), SOAPdenovo (Li *et al.*, 2010) or SGA (Simpson and Durbin, 2012), include their own correcting mechanisms.

The importance of the problem led to the design of many stand-alone error-correcting programs, such as Euler (Chaisson *et al.*, 2004), SHREC (Schröder *et al.*, 2009), Reptile (Yang *et al.*, 2010), Quake (Kelley *et al.*, 2010), CUDA (Shi *et al.*, 2010), HSHREC (Salmela, 2010), SOAP (Li *et al.*, 2010), HiTEC (Ilie *et al.*, 2011), Coral (Salmela and Schröder, 2011), Hammer (Medvedev *et al.*, 2011), ECHO (Kao *et al.*, 2011), PSAEC (Zhao *et al.*, 2011b) and MyHybrid (Zhao *et al.*, 2011a); see also the survey of Yang *et al.* (2013).

The errors present in sequencing data consist of substitutions, where the correct base has been replaced by an erroneous one, and indels, where new bases have been inserted or existing ones deleted. However, due to the domination of the Illumina technology, which produces mostly substitution errors, the majority of the existing programs focus on correcting this type of errors. For the same reason, the current version of our new program focuses as well on substitution errors.

The main idea of all correcting programs is essentially the same. High-throughput technologies compensate for short read length by high coverage, which implies that each position in the genome is sequenced multiple times, many of which are correct. Therefore, the correct way will prevail in a careful analysis and the errors can be corrected. It is the way in which such an analysis is done that makes the difference between various programs.

Some programs count the number of k-mers and those with counts above a given threshold are deemed correct, whereas the remaining ones undergo correction; these include SHREC, HiTEC, HSHREC and PSAEC. Others consider all k-mers occurring in each read, the k-spectrum (Pevzner *et al.*, 2001), and try to correct the entire read with minimum edit distance such that all k-mers have counts above a threshold; CUDA, Quake, Reptile and Hammer are included here. Finally, Coral, ECHO and MyHybrid are based on multiple sequence alignments. The reader is referred to the survey of Yang *et al.* (2013) for details.

We propose a new program, RACER (Rapid and Accurate Correction of Errors in Reads), that belongs to the first category. Although SHREC and HSHREC use suffix trees and HiTEC uses the more efficient suffix arrays, RACER uses completely different, more efficient data structures. This way, whereas both SHREC and, to a lesser extent, HiTEC, have high space

*To whom correspondence should be addressed.

Table 1. The datasets used for evaluation, sorted increasingly by total number of base pairs

Dataset	Accession number	Reference genome	Genome length	Read length	Number of reads	Number of base pairs	Coverage
Lactococcus lactis	SRR088759	NC_013656.1	2 598 144	36	4 370 050	157 321 800	60.55
Treponema pallidum	SRR361468	CP002376.1	1 139 417	35	7 133 663	249 678 205	219.13
E.coli 75a	SRR396536	NC_000913.2	4 639 675	75	3 454 048	259 053 600	55.83
Bacillus subtilis	DRR000852	NC_000964.3	4 215 606	75	3 519 504	263 962 800	62.62
E.coli 75b	SRR396532	NC_000913.2	4 639 675	75	4 341 061	325 579 575	70.17
Pseudomonas aeruginosa	SRR396641	NC_002516.2	6 264 404	36	9 306 557	335 036 052	53.48
E.coli 47	SRR022918	NC_000913.1	4 771 872	47	14 408 630	677 205 610	141.92
Leptospira interrogans L	SRR353563	NC_004342.2	4 338 762	100	7 066 162	706 616 200	162.86
L.interrogans C	SRR397962	NC_005823.1	4 277 185	100	7 127 250	712 725 000	166.63
E.coli 36	SRX000429	NC_000913.1	4 771 872	36	20 816 448	749 392 128	157.04
Haemophilus influenzae	SRR065202	NC_000907.1	1 830 138	42	23 935 272	1 005 281 424	549.29
S.aureus	SRR022866	NC_003923.1	2 901 156	76	25 551 716	1 941 930 416	669.36
S.cerevisiae	SRX100885	PRJNA128	12 416 363	76	52 061 664	3 956 686 464	318.67
C.elegans	SRR065390	wormbase.org	102 291 899	100	67 617 092	6 761 709 200	66.10
D.melanogaster	SRX006151 SRX006152 SRX023452	flybase.org	120 220 296	45 75 95	101 548 652	6 903 898 304	57.43

Note: All datasets and reference genome sequences are obtained from National Center for Biotechnology Information except *C.elegans* from www.wormbase.org and *D.melanogaster* from flybase.org.

consumption, RACER is space efficient. It is also faster and more effective at correcting errors than the existing programs. We have performed extensive comparison with the leading programs on a variety of real datasets. RACER is freely available for non-commercial use at www.csd.uwo.ca/∼ilie/RACER/.

2 RESULTS

2.1 Datasets

We have performed extensive testing on a wide variety of datasets with difference read length, genome size and coverage. Comparison has been performed exclusively on real datasets because, in our experience, simulated datasets do not offer a good indication of real-life performance. In addition, they are easy to correct and the correcting programs often exhibit unrealistically high performance.

Fifteen real datasets were considered, some of which have been previously used either for correcting purposes or assembly: *Escherichia coli 36*, *E.coli 47* (the numbers following the name represent read length and are used to distinguish them), *Staphylococcus aureus*, *Saccharomyces cerevisiae* and *Drosophila melanogaster* have been used in the survey of Yang *et al*. (2013), and *Caenorhabditis elegans* was used for comparing SGA with other genome assemblers by Simpson and Durbin (2012). The remaining datasets are new. The accession numbers and details of the datasets, together with the corresponding information concerning the reference genomes, are given in Table 1. Full organism names are provided in Supplementary Table S4.

2.2 Evaluation

We have compared RACER with the programs tested by Yang *et al*. (2013), namely, HiTEC, SHREC, Reptile, Quake, Coral, SOAP and ECHO. (We have tested SHREC instead of HSHREC, as we consider only substitution errors.) Of these, SOAP and ECHO were unable to run most of the datasets because of high space or time requirements and are left out of the comparison.

All programs tested were evaluated on the raw datasets in Table 1 for their ability to correct errors as well as the time and space required. Error-correcting performance was evaluated as the percentage of errors corrected. Our evaluation method has the advantage of avoiding the interference of mapping or assembling programs and is described in detail in Section 3.2 of Methods.

To have a uniform comparison, we have normalized the time and space values by dividing them by the total number of base pairs in the dataset. The actual time and space values are given in the Supplementary Material.

We have run all programs on the same Hewlett–Packard computer with 24 cores AMD Opteron at 2.1 GHz and 98 GB of random access memory running Linux Red Hat, CentOS 5.5 m. Because HiTEC and Reptile do not run in parallel, we performed the testing in both serial and parallel modes, to include all programs in the comparison. We have used 24 cores in parallel mode.

The results are presented in Table 2, which contains the error-correcting performance in the top part in percentage of errors corrected, the time in the middle part in seconds per megabyte of input base pairs and space in the bottom part in megabytes required per megabyte of input base pairs. Some programs could not run all datasets and the reasons are indicated by letters explained in the caption of the table. The last row in each part represents the average of the entries for those datasets for which all programs could run. To improve the visualization of the tables, we have used color intensities with darker color representing better performance.

Table. 2. Comparison of the error correction programs on the datasets in Table 1, in serial and parallel (24 cores) modes, with respect to error-correcting performance, time and space usage

	Serial						Parallel (24 cores)			
	Coral	HiTEC	Quake	Reptile	SHREC	RACER	Coral	Quake	SHREC	RACER
Error correction										
L.lactis	65.54	80.61	71.65	60.27	(b)	80.49	65.52	71.31	(b)	80.49
T.pallidum	38.55	84.45	59.46	2.65	61.79	85.75	38.54	59.24	61.50	85.75
E.coli 75a	26.04	82.72	1.50	21.82	72.61	83.58	26.04	2.11	71.67	83.58
B.subtilis	59.76	80.59	53.59	64.25	41.19	82.12	59.77	53.63	40.54	82.12
E.coli 75b	9.80	76.38	2.51	54.55	38.58	76.32	9.80	2.18	37.78	76.32
P.aeruginosa	79.78	78.68	7.08	68.44	63.40	85.32	79.77	30.48	63.37	85.32
E.coli 47	0.00	19.35	8.53	0.00	(b)	56.50	0.00	8.46	(b)	56.50
L.interrogans L	48.25	60.23	49.75	35.55	55.99	59.87	48.25	49.75	55.15	59.87
L.interrogans C	44.16	54.26	44.97	38.46	48.09	53.91	44.16	44.95	47.39	53.91
E.coli 36	58.02	85.89	81.38	0.06	77.49	86.32	58.02	81.43	77.03	86.32
H.influenzae	28.39	73.33	60.52	10.64	53.45	78.35	28.39	(d)	53.19	78.35
S.aureus	0.02	0.03	(d)	0.03	(a)	25.96	0.02	(d)	(a)	25.96
S.cerevisiae	2.85	0.23	6.81	11.38	9.17	12.25	2.85	6.90	8.96	12.25
C.elegans	(a)	(a)	38.88	0.21	(a)	56.54	(a)	38.87	(a)	56.54
D.melanogaster	(a)	(c)	35.36	0.56	(a)	42.95	(a)	35.47	(a)	42.95
Average	45.54	75.40	37.53	35.72	57.39	76.65	45.54	40.47	56.80	76.65
Time (s/MB)										
L.lactis	11.60	5.68	11.29	4.15	(b)	1.16	1.39	3.73	(b)	0.15
T.pallidum	30.76	11.07	18.10	9.52	22.57	3.28	2.24	4.32	2.30	0.30
E.coli 75a	25.62	12.44	3.41	9.10	27.01	5.53	2.07	2.66	3.19	0.40
B.subtilis	24.60	12.37	5.20	7.90	25.11	4.24	2.04	2.77	3.05	0.44
E.coli 75b	27.14	12.89	3.57	10.65	27.39	4.26	2.20	2.34	4.34	0.38
P.aeruginosa	11.46	2.87	4.22	14.85	17.69	1.28	1.33	3.13	2.06	0.20
E.coli 47	6.13	12.17	35.42	6.98	(b)	4.32	1.14	5.57	(b)	0.75
L.interrogans L	95.48	11.63	2.49	4.95	26.88	3.37	5.13	1.26	3.65	0.27
L.interrogans C	101.94	12.99	2.71	4.84	27.55	2.99	5.37	1.73	3.05	0.24
E.coli 36	26.77	13.45	14.16	6.01	19.14	2.12	2.09	3.12	2.25	0.28
H.influenzae	88.37	14.15	6.37	10.75	19.54	2.22	4.47	(d)	2.22	0.28
S.aureus	76.80	2.03	(d)	15.93	(a)	4.60	4.71	(d)	(a)	0.58
S.cerevisiae	95.17	2.14	2.36	7.73	26.61	3.64	5.01	1.28	5.36	0.34
C.elegans	(a)	(a)	3.10	16.13	(a)	5.30	(a)	1.07	(a)	0.41
D.melanogaster	(a)	(c)	10.59	19.59	(a)	7.06	(a)	3.70	(a)	0.95
Average	42.97	11.21	6.73	8.48	24.17	3.38	2.81	2.67	2.99	0.31
Space (MB/MB)										
L.lactis	13.30	19.86	3.29	3.86	(b)	3.43	350.21	10.79	(b)	4.81
T.pallidum	13.74	19.90	5.45	3.23	147.74	2.39	226.02	7.72	417.84	3.35
E.coli 75a	15.42	19.81	4.98	7.14	145.67	5.82	220.01	7.87	402.32	7.18
B.subtilis	16.86	19.79	5.29	6.34	144.77	5.71	217.65	7.73	395.65	7.04
E.coli 75b	15.81	19.53	5.58	11.59	117.41	4.76	178.51	7.02	320.80	5.81
P.aeruginosa	11.20	19.84	5.79	2.47	112.25	3.65	169.60	8.06	311.49	4.47
E.coli 47	18.48	19.75	5.21	2.96	(b)	9.96	96.70	6.26	(b)	10.75
L.interrogans L	12.18	10.76	3.97	3.61	56.43	1.43	87.15	4.79	147.84	2.05
L.interrogans C	11.74	10.80	4.12	3.42	55.85	1.42	86.06	4.62	146.52	1.84
E.coli 36	11.52	19.84	5.72	1.50	52.99	1.97	82.30	7.01	139.24	2.73
H.influenzae	10.67	19.96	4.44	1.11	39.78	1.33	63.43	(d)	103.93	1.74
S.aureus	23.75	19.50	(d)	4.77	(a)	2.46	51.03	(d)	(a)	2.76
S.cerevisiae	10.94	20.64	3.99	1.17	18.32	1.49	24.34	4.13	26.50	1.66
C.elegans	(a)	(a)	4.96	1.61	(a)	2.76	(a)	5.07	(a)	2.83
D.melanogaster	(a)	(c)	5.52	3.20	(a)	6.26	(a)	5.60	(a)	6.41
Average	13.56	17.53	5.11	4.91	104.14	3.39	158.41	6.85	285.21	4.31

Note: Letters are used in place of numbers whenever a program failed to run a dataset as follows: (a) insufficient memory, (b) java.lang.NegativeArraySizeException, (c) reads of different length exist and (d) *k*-mer cutoff failed. The average in the bottom row of each section includes only the datasets run by all programs.

2.3 Comparison

As far as error-correcting performance is concerned, only HiTEC comes close to RACER. However, HiTEC requires the second highest space and, therefore, cannot run the largest datasets. For *D.melanogaster*, HiTEC did not attempt to run it because of different read lengths. Note also HiTEC's poor performance on some of the datasets that were not included in the average. Direct comparison shows a significant difference.

Concerning time, RACER was the fastest in both serial and parallel modes, twice faster than second-placed Quake in serial and about one order of magnitude faster than all the others in parallel mode. RACER is also more memory efficient than the other programs. Quake is again second best and uses >50% more space than RACER in parallel mode.

RACER and Reptile were the only programs able to run all datasets. Quake came second in time and space, except serial space where it is third, but its error-correcting performance was second last in serial and last in parallel, about half of the performance of RACER. This is in part because Quake trims reads when unable to correct. The evaluation counts the number of base pairs in all correct reads. Reptile came third in time and second in space (serial mode) but its accuracy was last. SHREC required clearly the highest space and for that reason it was able to run the fewest datasets on our machine. On the other hand, it came consistently in third place for correcting performance. Coral was the slowest in serial mode but had the best speedup among all programs, and its parallel performance was close to that of Quake and SHREC. The 15× speedup came at the cost of increasing the space 12 times. RACER's speedup was 11× with a small increase in space.

3 METHODS

3.1 RACER

RACER belongs to the class of *k*-mer counting programs. It uses 2-bit encoding of nucleotides and random replacement of the unknown positions. Each *k*-mer is represented as a 64-bit integer. The *k*-mers are stored in a hash table. For each *k*-mer, eight counters, corresponding to the eight possible nucleotides on both sides, are computed. A threshold t is used to distinguish correct from erroneous positions. A *k*-mer followed by a nucleotide a is assumed correct if the count is higher than t and erroneous otherwise. In the latter case, assuming there is exactly one letter b that follows the same *k*-mer and has count higher than t, a is replaced by b.

The approximate size of the sequenced genome is required as input, from which RACER automatically computes the *k*-mer size k and threshold t used in the correction part. The combination of parameters used has been experimentally determined. Like HiTEC, the thresholds are varied to achieve higher correcting performance.

A significant space advantage over SHREC and HiTEC is given by the avoidance of full text indexes, such as suffix trees and suffix arrays. The same reason creates a speed advantage because the *k*-mers and counters, once computed, can be used multiple times for correction.

RACER corrects reads of varying length, from both fasta and fastq data. It has been implemented in C++ and OpenMP and no other software is necessary to run it.

3.2 Evaluation details

To obtain the most accurate evaluation, we have avoided the use of mapping programs or performance of corrected data for other applications, such as genome assembly. Mapping programs are used for evaluation to know where the errors are and how they should be corrected. However, many reads cannot be mapped uniquely and even more cannot be mapped at all. Such reads are discarded and, therefore, the obtained datasets are significantly different from the original. In particular, all programs have artificially increased performance because the reads that are the most difficult to correct have been eliminated by mapping. On the other hand, genome assemblers have their own implicit or explicit correction procedure, and the interaction between the two correction methods cannot be predicted.

Reads are classified as correct or erroneous depending on whether they can be found or not, respectively, in the reference genome by an exact search algorithm. Denoting the number of base pairs in erroneous reads before and after correction by e_b and e_a, respectively, the error-correcting

performance was computed as $(e_b - e_a)/e_b$. Because we have $e_b = TP + FN$, $e_a = FP + FN$ (Ilie *et al.*, 2011), we have that $\frac{e_b - e_a}{e_b} = \frac{TP - FP}{TP + FN}$, which is the same as the formula used by Yang *et al.* (2013). By considering whole-read correction, as explained above, instead of individual-based correction, similar performance is expected; however, the problems associated with read mapping are avoided.

Nevertheless, because in some articles the comparison is performed on mapped datasets, we provide, for completeness, such a comparison in the Supplementary Material where the datasets have been mapped using BWA (Li and Durbin, 2009) with high error rate, so that a high number of reads are kept and bias is minimized. The comparison on the mapped datasets is similar with the one on the raw datasets except that, as expected, the percentage of errors corrected increases.

All programs have been run according to the instructions given by the authors in the corresponding articles, Web sites or readme files. We have not tuned the parameters of any of the programs to improve performance, as this would be unrealistic. The commands used to run all programs are given in the Supplementary Material.

4 DISCUSSION

Extensive testing shows that RACER is superior to the existing programs in all aspects: error-correcting performance, time and space. It corrects about three-quarters of the errors in <2 min for a bacterial dataset and 30–40 min for a larger organism such as a worm or fly. RACER does not require any additional software to run.

5 CONCLUSION

We have presented a new tool, RACER, for correcting errors in NGS data. The current version of RACER targets Illumina data, thus correcting substitution error. Future versions will be able to handle indel errors to enable correction of data from other sequencing platforms as well as mixed data.

ACKNOWLEDGEMENTS

Performance evaluation has been performed using the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET: www.sharcnet.ca) and Compute/Calcul Canada.

Funding: The Natural Sciences and Engineering Research Council of Canada (NSERC) (in parts to L.I.).

Conflict of Interest: none declared.

REFERENCES

Butler,J. *et al.* (2008) ALLPATHS: *de novo* assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.

Chaisson,M. *et al.* (2004) Fragment assembly with short reads. *Bioinformatics*, **20**, 2067–2074.

Chaisson,M.J. and Pevzner,P.A. (2008) Short read fragment assembly of bacterial genomes. *Genome Res.*, **18**, 324–330.

Haussler,D. *et al.* (2009) Genome 10K: a proposal to obtain whole-genome sequence for 10,000 vertebrate species. *J. Hered.*, **100**, 659–674.

Ilie,L. *et al.* (2011) HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics*, **27**, 295–302.

Kao,W.C. *et al.* (2011) ECHO: a reference-free short-read error correction algorithm. *Genome Res.*, **21**, 1181–1192.

Kelley,D.R. *et al.* (2010) Quake: quality-aware detection and correction of sequencing errors. *Genome Biol.*, **11**, R116.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li,R. *et al.* (2010) *De novo* assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, **20**, 265–272.

Medvedev,P. *et al.* (2011) Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics*, **27**, i137–i141.

Metzker,M.L. (2010) Sequencing technologies - the next generation. *Nat. Rev. Genet.*, **11**, 31–46.

Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl Acad. Sci. USA*, **98**, 9748–9753.

Salmela,L. (2010) Correction of sequencing errors in a mixed set of reads. *Bioinformatics*, **26**, 1284–1290.

Salmela,L. and Schröder,J. (2011) Correcting errors in short reads by multiple alignments. *Bioinformatics*, **27**, 1455–1461.

Sanger,F. *et al.* (1977) DNA sequencing with chain-terminating inhibitors. *Proc. Natl Acad. Sci. USA*, **74**, 5463–5467.

Schröder,J. *et al.* (2009) SHREC: a short-read error correction method. *Bioinformatics*, **25**, 2157–2163.

Shi,H. *et al.* (2010) A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. *J. Comput. Biol.*, **17**, 603–615.

Simpson,J.T. and Durbin,R. (2012) Efficient *de novo* assembly of large genomes using compressed data structures. *Genome Res.*, **22**, 549–556.

Simpson,J.T. *et al.* (2009) ABySS: a parallel assembler for short read sequence data. *Genome Res.*, **19**, 1117–1123.

Siva,N. (2008) 1000 Genomes Project. *Nat. Biotechnol.*, **26**, 256.

Yang,X. *et al.* (2010) Reptile: representative tiling for short read error correction. *Bioinformatics*, **26**, 2526–2533.

Yang,X. *et al.* (2013) A survey of error-correction methods for next-generation sequencing. *Brief. Bioinform.*, **14**, 56–66.

Zhao,Z. *et al.* (2011a) An efficient hybrid approach to correcting errors in short reads. In: Torra,V., Narakawa,Y., Yin,J. and Long,J. (eds) *Modeling Decision for Artificial Intelligence*. Volume 6820 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pp. 98–210.

Zhao,Z. *et al.* (2011b) PSAEC: an improved algorithm for short read error correction using partial suffix arrays. In: Atallah,M., Li,X.Y. and Zhu,B. (eds) *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*. Volume 6681 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pp. 220–232.