BIBLIOGRAPHY

[1] Philip S. Abrams. *An APL machine*. PhD thesis, Stanford University, Stanford, CA, February 1970. (Technical Report SLAC-R-114, Stanford Linear Accelerator Center, Stanford University, February 1970.)

[2] Alfred V. Aho, Mahadevan Ganapathi, and Steven W. K. Tjiang. Code generation using tree matching and dynamic programming. *ACM Transactions on Programming Languages and Systems*, 11(4):491–516, October 1989.

[3] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. On finding lowest common ancestors in trees. In *Conference Record of the Fifth Annual ACM Symposium on Theory of Computing* (STOC), pages 253–265, May 1973.

[4] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[5] Alfred V. Aho and Stephen C. Johnson. Optimal code generation for expression trees. *Journal of the ACM*, 23(3):488–501, July 1976.

[6] Alfred V. Aho, Stephen C. Johnson, and Jeffrey D. Ullman. Code generation for expressions with common subexpressions. In *Conference Record of the Third ACM Symposium on Principles of Programming Languages*, pages 19–31, Atlanta, GA, January 1976.

[7] Alfred V. Aho, Steven C. Johnson, and Jeffrey D. Ullman. Deterministic parsing of ambiguous grammars. In *Conference Record of the ACM Symposium on Principles of Programming Languages*, pages 1–21, Boston, MA, October 1973.

[8] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA, 1986.

[9] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[10] Philippe Aigrain, Susan L. Graham, Robert R. Henry, Marshall Kirk McKusick, and Eduardo Pelegri-Llopart. Experience with a Graham-Glanville style code generator. *SIGPLAN Notices*, 19(6):13–24, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*.

[11] Alexander Aiken and Alexandru Nicolau. Optimal loop parallelization. *SIGPLAN Notices*, 23(7):308–317, July 1988. *Proceedings of the ACM SIGPLAN '88 Conference on Programming Language Design and Implementation*.

[12] Frances E. Allen. Program optimization. In *Annual Review in Automatic Programming*, volume 5, pages 239–307. Pergamon Press, Oxford, England, 1969.

[13] Frances E. Allen. Control flow analysis. *SIGPLAN Notices*, 5(7):1–19, July 1970. *Proceedings of a Symposium on Compiler Optimization*.

[14] Frances E. Allen. The history of language processor technology in IBM. *IBM Journal of Research and Development*, 25(5):535–548, September 1981.

[15] Frances E. Allen and John Cocke. A catalogue of optimizing transformations. In R. Rustin, editor, *Design and Optimization of Compilers*, pages 1–30. Prentice-Hall, Englewood Cliffs, NJ, June 1972.

[16] Frances E. Allen and John Cocke. Graph-theoretic constructs for program flow analysis. Technical Report RC 3923 (17789), IBM Thomas J. Watson Research Center, Yorktown Heights, NY, July 1972.

[17] Frances E. Allen and John Cocke. A program data flow analysis procedure. *Communications of the ACM*, 19(3):137–147, March 1976.

[18] Frances E. Allen, John Cocke, and Ken Kennedy. Reduction of operator strength. In Steven S.Muchnick and Neil D. Jones, editors, *Program Flow Analysis: Theory and Applications*, pages 79–101. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[19] John R. Allen and Ken Kennedy. *Optimizing Compilers for Modern Architectures*. Morgan Kaufmann, San Francisco, CA, October 2001.

[20] Bowen Alpern and Fred B. Schneider. Verifying temporal properties without temporal logic. *ACM Transactions on Programming Languages and Systems*, 11(1):147–167, January 1989.

[21] Bowen Alpern, Mark N. Wegman, and F. Kenneth Zadeck. Detecting equality of variables in programs. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages*, pages 1–11, San Diego, CA, January 1988.

[22] Stephen Alstrup, Dov Harel, Peter W. Lauridsen, and Mikkel Thorup. Dominators in linear time. *SIAM Journal on Computing*, 28(6):2117–2132, June 1999.

[23] Marc A. Auslander and Martin E. Hopkins. An overview of the PL.8 compiler. *SIGPLAN Notices*, 17(6):22–31, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction*.

[24] Andrew Ayers, Robert Gottlieb, and Richard Schooler. Aggressive inlining. *SIGPLAN Notices*, 32(5):134–145, May 1997. *Proceedings of the ACM SIGPLAN '97 Conference on Programming Language Design and Implementation*.

[25] John W. Backus. The history of Fortran I, II, and III. In Richard L. Wexelblat, editor, *History of Programming Languages*, pages 25–45. Academic Press, New York, NY, 1981.

[26] John W. Backus, R. J. Beeber, S. Best, R. Goldberg, L. M. Haibt, H. L. Herrick, R. A. Nelson, D. Sayre, P. B. Sheridan, H. Stern, I. Ziller, R. A. Hughes, and R. Nutt. The FORTRAN automatic coding system. In *Proceedings of the Western Joint Computer Conference*, pages 188–198, Institute of Radio Engineers, New York, NY, February 1957.

[27] David F. Bacon, Susan L. Graham, and Oliver J. Sharp. Compiler transformations for high-performance computing. *ACM Computing Surveys*, 26(4):345–420, 1994.

[28] John T. Bagwell, Jr. Local optimizations. *SIGPLAN Notices*, 5(7):52–66, July 1970. *Proceedings of a Symposium on Compiler Optimization*.

[29] John Banning. An efficient way to find side effects of procedure calls and aliases of variables. In *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 29–41, San Antonio, TX, January 1979.

[30] William A. Barrett and John D. Couch. *Compiler Construction: Theory and Practice.* Science Research Associates, Inc., Chicago, IL, 1979.

[31] Jeffrey M. Barth. An interprocedural data flow analysis algorithm. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages*, pages 119–131, Los Angeles, CA, January 1977.

[32] Alan M. Bauer and Harry J. Saal. Does *APL* really need run-time checking? *Software—Practice and Experience*, 4(2):129–138, 1974.

[33] Laszlo A. Belady. A study of replacement algorithms for a virtual storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

[34] C. Gordon Bell and Allen Newell. *Computer Structures: Readings and Examples.* McGraw-Hill Book Company, New York, NY, 1971.

[35] Peter Bergner, Peter Dahl, David Engebretsen, and Matthew T. O'Keefe. Spill code minimization via interference region spilling. *SIGPLAN Notices*, 32(5):287–295, May 1997. *Proceedings of the ACM SIGPLAN '97 Conference on Programming Language Design and Implementation.*

[36] David Bernstein, Dina Q. Goldin, Martin Charles Golumbic, Hugo Krawczyk, Yishay Mansour, Itai Nahshon, and Ron Y. Pinter. Spill code minimization techniques for optimizing compilers. *SIGPLAN Notices*, 24(7):258–263, July 1989. *Proceedings of the ACM SIGPLAN '89 Conference on Programming Language Design and Implementation.*

[37] David Bernstein and Michael Rodeh. Global instruction scheduling for superscalar machines. *SIGPLAN Notices*, 26(6):241–255, June 1991. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation.*

[38] Robert L. Bernstein. Producing good code for the case statement. *Software—Practice and Experience*, 15(10):1021–1024, October 1985.

[39] Andrew Binstock and John Rex. *Practical Algorithms for Programmers.* Addison-Wesley, Reading, MA, 1995.

[40] Peter L. Bird. An implementation of a code generator specification language for table driven code generators. *SIGPLAN Notices*, 17(6):44–55, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction.*

[41] Rastislav Bodík, Rajiv Gupta, and Mary Lou Soffa. Complete removal of redundant expressions. *SIGPLAN Notices*, 33(5):1–14, May 1998. *Proceedings of the ACM SIGPLAN '98 Conference on Programming Language Design and Implementation.*

[42] Hans-Juergen Boehm. Space efficient conservative garbage collection. *SIGPLAN Notices*, 28(6):197–206, June 1993. *Proceedings of the ACM SIGPLAN '93 Conference on Programming Language Design and Implementation.*

[43] Hans-Juergen Boehm and Alan Demers. Implementing Russell. *SIGPLAN Notices*, 21(7):186–195, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction.*

[44] Hans-Juergen Boehm and Mark Weiser. Garbage collection in an uncooperative environment. *Software—Practice and Experience*, 18(9):807–820, September 1988.

[45] David G. Bradlee, Susan J. Eggers, and Robert R. Henry. Integrating register allocation and instruction scheduling for RISCS. *SIGPLAN Notices*, 26(4):122–131, April 1991. *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Systems.*

[46] Preston Briggs. *Register Allocation via Graph Coloring*. PhD thesis, Rice University, Department of Computer Science, Houston, TX, April 1992. (Technical Report TR92–183, Computer Science Department, Rice University, 1992.)

[47] Preston Briggs, Keith D. Cooper, Timothy J. Harvey, and L. Taylor Simpson. Practical improvements to the construction and destruction of static single assignment form. *Software—Practice and Experience*, 28(8):859–881, July 1998.

[48] Preston Briggs, Keith D. Cooper, Ken Kennedy, and Linda Torczon. Coloring heuristics for register allocation. *SIGPLAN Notices*, 24(7):275–284, July 1989. *Proceedings of the ACM SIGPLAN '89 Conference on Programming Language Design and Implementation*.

[49] Preston Briggs, Keith D. Cooper, Ken Kennedy, and Linda Torczon. Digital computer register allocation and code spilling using interference graph coloring. United States Patent 5,249,295, March 1993.

[50] Preston Briggs, Keith D. Cooper, and L. Taylor Simpson. Value numbering. *Software—Practice and Experience*, 27(6):701–724, June 1997.

[51] Preston Briggs, Keith D. Cooper, and Linda Torczon. Coloring register pairs. *ACM Letters on Programming Languages and Systems*, 1(1):3–13, March 1992.

[52] Preston Briggs, Keith D. Cooper, and Linda Torczon. Rematerialization. *SIGPLAN Notices*, 27(7):311–321, July 1992. *Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation*.

[53] Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to graph coloring register allocation. *ACM Transactions on Programming Languages and Systems*, 16(3):428–455, May 1994.

[54] Preston Briggs and Linda Torczon. An efficient representation for sparse sets. *ACM Letters on Programming Languages and Systems*, 2(1–4):59–69, March–December 1993.

[55] Adam L. Buchsbaum, Haim Kaplan, Anne Rogers, and Jeffery R. Westbrook. Linear-time pointer-machine algorithms for least common ancestors, *MST* verification, and dominators. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 279–288, Dallas, TX, 1998.

[56] Michael Burke. An interval-based approach to exhaustive and incremental interprocedural data-flow analysis. *ACM Transactions on Programming Languages and Systems*, 12(3):341–395, July 1990.

[57] Michael Burke and Linda Torczon. Interprocedural optimization: Eliminating unnecessary recompilation. *ACM Transactions on Programming Languages and Systems*, 15(3):367–399, July 1993.

[58] Jiazhen Cai and Robert Paige. Using multiset discrimination to solve language processing problems without hashing. *Theoretical Computer Science*, 145(1–2):189–228, 1995.

[59] Brad Calder and Dirk Grunwald. Reducing branch costs via branch alignment. *SIGPLAN Notices*, 29(11): 242–251, November 1994. *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*.

[60] David Callahan, Alan Carle, Mary W. Hall, and Ken Kennedy. Constructing the procedure call multigraph. *IEEE Transactions on Software Engineering*, 16(4):483–487, April 1990.

[61] David Callahan, Steve Carr, and Ken Kennedy. Improving register allocation for subscripted variables. *SIGPLAN Notices*, 25(6):53–65, June 1990. *Proceedings of the ACM SIGPLAN '90 Conference on Programming Language Design and Implementation.*

[62] David Callahan, Keith D. Cooper, Ken Kennedy, and Linda Torczon. Interprocedural constant propagation. *SIGPLAN Notices*, 21(7):152–161, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction.*

[63] David Callahan and Brian Koblenz. Register allocation via hierarchical graph coloring. *SIGPLAN Notices*, 26(6):192–203, June 1991. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation.*

[64] Luca Cardelli. Type systems. In Allen B. Tucker, Jr., editor, *The Computer Science and Engineering Handbook*, chapter 103, pages 2208–2236. CRC Press, Boca Raton, FL, December 1996.

[65] Steve Carr and Ken Kennedy. Scalar replacement in the presence of conditional control flow. *Software—Practice and Experience*, 24(1):51–77, 1994.

[66] Roderic G. G. Cattell. Automatic derivation of code generators from machine descriptions. *ACM Transactions on Programming Languages and Systems*, 2(2):173–190, April 1980.

[67] Roderic G. G. Cattell, Joseph M. Newcomer, and Bruce W. Leverett. Code generation in a machine-independent compiler. *SIGPLAN Notices*, 14(8):65–75, August 1979. *Proceedings of the ACM SIGPLAN '79 Symposium on Compiler Construction.*

[68] Gregory J. Chaitin. Register allocation and spilling via graph coloring. *SIGPLAN Notices*, 17(6):98–105, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction.*

[69] Gregory J. Chaitin. Register allocation and spilling via graph coloring. United States Patent 4,571,678, February 1986.

[70] Gregory J. Chaitin, Marc A. Auslander, Ashok K. Chandra, John Cocke, Martin E. Hopkins, and Peter W. Markstein. Register allocation via coloring. *Computer Languages*, 6(1):47–57, January 1981.

[71] David R. Chase. An improvement to bottom-up tree pattern matching. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages*, pages 168–177, Munich, Germany, January 1987.

[72] David R. Chase, Mark Wegman, and F. Kenneth Zadeck. Analysis of pointers and structures. *SIGPLAN Notices*, 25(6):296–310, June 1990. *Proceedings of the ACM SIGPLAN '90 Conference on Programming Language Design and Implementation.*

[73] J. Bradley Chen and Bradley D. D. Leupen. Improving instruction locality with just-in-time code layout. In *Proceedings of the First USENIX Windows NT Workshop*, pages 25–32, Seattle, WA, August 1997.

[74] C. J. Cheney. A nonrecursive list compacting algorithm. *Communications of the ACM*, 13(11):677–678, November 1970.

[75] Jong-Deok Choi, Michael Burke, and Paul R.Carini. Efficient flow-sensitive interprocedural computation of pointer-induced aliases and side effects. In *Conference Record of the Twentieth Annual ACM Symposium on Principles of Programming Languages*, pages 232–245, Charleston, SC, January 1993.

[76] Frederick C. Chow. *A Portable Machine-Independent Global Optimizer —Design and Measurements*. PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, December 1983. (Technical Report CSL-TR-83-254, Computer Systems Laboratory, Stanford University, December 1983.)

[77] Frederick C. Chow and John L. Hennessy. Register allocation by priority-based coloring. *SIGPLAN Notices*, 19(6):222–232, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction.*

[78] Frederick C. Chow and John L. Hennessy. The priority-based coloring approach to register allocation. ACM *Transactions on Programming Languages and Systems*, 12(4):501–536, October 1990.

[79] Cliff Click. *Combining Analyses, Combining Optimizations*. PhD thesis, Rice University, Department of Computer Science, Houston, TX, February 1995. (Technical Report TR95-252, Computer Science Department, Rice University, 1995.)

[80] Cliff Click. Global code motion/global value numbering. *SIGPLAN Notices*, 30(6):246–257, June 1995. *Proceedings of the ACM SIGPLAN '95 Conference on Programming Language Design and Implementation.*

[81] Cliff Click and Keith D. Cooper. Combining analyses, combining optimizations. *ACM Transactions on Programming Languages and Systems*, 17(2):181–196, 1995.

[82] John Cocke. Global common subexpression elimination. *SIGPLAN Notices*, 5(7):20–24, July 1970. *Proceedings of a Symposium on Compiler Construction.*

[83] John Cocke and Ken Kennedy. An algorithm for reduction of operator strength. *Communications of the ACM*, 20(11):850–856, November 1977.

[84] John Cocke and Peter W. Markstein. Measurement of program improvement algorithms. In Simon H. Lavington, editor, *Information Processing 80*, North Holland, Amsterdam, Netherlands, pages 221–228, 1980, *Proceedings of IFIP Congress 80.*

[85] John Cocke and Peter W. Markstein. Strength reduction for division and modulo with application to accessing a multilevel store. *IBM Journal of Research and Development*, 24(6):692–694, 1980.

[86] John Cocke and Jacob T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, New York, NY, 1970.

[87] Jacques Cohen. Garbage collection of linked structures. *ACM Computing Surveys*, 13(3):341–367, September 1981.

[88] Robert Cohn and P. Geoffrey Lowney. Hot cold optimization of large Windows/*NT* applications. In *Proceedings of the Twenty-Ninth Annual International Symposium on Microarchitecture*, pages 80–89, Paris, France, December 1996.

[89] Stephanie Coleman and Kathryn S. McKinley. Tile size selection using cache organization and data layout. *SIGPLAN Notices*, 30(6):279–290, June 1995. *Proceedings of the ACM SIGPLAN '95 Conference on Programming Language Design and Implementation.*

[90] George E. Collins. A method for overlapping and erasure of lists. *Communications of the ACM*, 3(12):655–657, December 1960.

[91] Melvin E. Conway. Design of a separable transition diagram compiler. *Communications of the ACM*, 6(7):396–408, July 1963.

[92] Richard W. Conway and Thomas R. Wilcox. Design and implementation of a diagnostic compiler for PL/I. *Communications of the ACM*, 16(3):169–179, March 1973.

[93] Keith D. Cooper, Mary W. Hall, and Linda Torczon. An experiment with inline substitution. *Software—Practice and Experience*, 21(6):581–601, June 1991.

[94] Keith D. Cooper, Timothy J. Harvey, and Linda Torczon. How to build an interference graph. *Software—Practice and Experience*, 28(4):425–444, April 1998.

[95] Keith D. Cooper, Timothy J. Harvey, and Todd Waterman. Building a control-flow graph from scheduled assembly code. Technical Report 02-399, Department of Computer Science, Rice University, Houston, TX, June 2002.

[96] Keith D. Cooper and Ken Kennedy. Interprocedural side-effect analysis in linear time. *SIGPLAN Notices*, 23(7):57–66, July 1988. *Proceedings of the ACM SIGPLAN '88 Conference on Programming Language Design and Implementation.*

[97] Keith D. Cooper and Ken Kennedy. Fast interprocedural alias analysis. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages*, pages 49–59, Austin, TX, January 1989.

[98] Keith D. Cooper and Philip J. Schielke. Non-local instruction scheduling with limited code growth. In *Proceedings of the 1998 ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems* (LCTES). *Lecture Notes in Computer Science 1474*, F. Mueller and A. Bestavros, editors, pages 193–207, Springer-Verlag, Heidelberg, Germany, 1998.

[99] Keith D. Cooper, Philip J. Schielke, and Devika Subramanian. Optimizing for reduced code space using genetic algorithms. *SIGPLAN Notices*, 34(7):1–9, July 1999. *Proceedings of the ACM SIGPLAN 1999 Workshop on Languages, Compilers, and Tools for Embedded Systems* (LCTES), May 1999.

[100] Keith D. Cooper and L. Taylor Simpson. Live range splitting in a graph coloring register allocator. In *Proceedings of the Seventh International Compiler Construction Conference, CC '98. Lecture Notes in Computer Science 1383*, pages 174–187, Springer-Verlag, Heidelberg, Germany, 1998.

[101] Keith D. Cooper, L. Taylor Simpson, and Christopher A. Vick. Operator strength reduction. *ACM Transactions on Programming Languages and Systems*, 23(5):603–625, September 2001.

[102] Keith D. Cooper and Todd Waterman. Understanding energy consumption on the C62x. In *Proceedings of the 2002 Workshop on Compilers and Operating Systems for Low Power*, pages 4–1 – 4–8, Charlottesville, VA, September 2002.

[103] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1992.

[104] Ron Cytron, Jeanne Ferrante, Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck. Efficiently computing static single assignment form and the control dependence graph. *ACM Transactions on Programming Languages and Systems*, 13(4):451–490, October 1991.

[105] Ron Cytron, Andy Lowry, and F. Kenneth Zadeck. Code motion of control structures in high-level languages. In *Conference Record of the Thirteenth Annual ACM Symposium on Principles of Programming Languages*, pages 70–85, St. Petersburg Beach, FL, January 1986.

[106] Manuvir Das. Unification-based pointer analysis with directional assignments. *SIGPLAN Notices*, 35(5):35–46, May 2000. In *Proceedings of the ACM SIGPLAN '00 Conference on Programming Language Design and Implementation*.

[107] Jack W. Davidson and Christopher W. Fraser. The design and application of a retargetable peephole optimizer. *ACM Transactions on Programming Languages and Systems*, 2(2):191–202, April 1980.

[108] Jack W. Davidson and Christopher W. Fraser. Automatic generation of peephole optimizations. *SIGPLAN Notices*, 19(6):111–116, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*.

[109] Jack W. Davidson and Christopher W. Fraser. Register allocation and exhaustive peephole optimization. *Software—Practice and Experience*, 14(9):857–865, September 1984.

[110] Jack W. Davidson and Christopher W. Fraser. Automatic inference and fast interpretation of peephole optimization rules. *Software—Practice and Experience*, 17(11):801–812, November 1987.

[111] Jack W. Davidson and Ann M. Holler. A study of a C function inliner. *Software—Practice and Experience*, 18(8):775–790, August 1988.

[112] Alan J. Demers, Mark Weiser, Barry Hayes, Hans Boehm, Daniel Bobrow, and Scott Shenker. Combining generational and conservative garbage collection: Framework and implementations. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 261–269, San Francisco, CA, January 1990.

[113] Frank DeRemer. Simple LR(k) grammars. *Communications of the* ACM, 14(7):453–460, July 1971.

[114] Frank DeRemer and Thomas J. Pennello. Efficient computation of LALR (1) look-ahead sets. *SIGPLAN Notices*, 14(8):176–187, August 1979. *Proceedings of the ACM SIGPLAN '79 Symposium on Compiler Construction*.

[115] Alain Deutsch. Interprocedural May-Alias analysis for pointers: Beyond k-limiting. *SIGPLAN Notices*, 29(6):230–241, June 1994. *Proceedings of the ACM SIGPLAN '94 Conference on Programming Language Design and Implementation*.

[116] L. Peter Deutsch. *An Interactive Program Verifier*. PhD thesis, Computer Science Department, University of California, Berkeley, Berkeley, CA, 1973. (Technical Report CSL-73-1, Xerox Palo Alto Research, May 1973.)

[117] L. Peter Deutsch and Daniel G Bobrow. An efficient, incremental, automatic, garbage collector. *Communications of the ACM*, 19(9):522–526, September 1976.

[118] Dhananjay M. Dhamdhere. On algorithms for operator strength reduction. *Communications of the ACM*, 22(5):311–312, May 1979.

[119] Dhananjay M. Dhamdhere. A fast algorithm for code movement optimisation. *SIGPLAN Notices*, 23(10):172–180, 1988.

[120] Dhananjay M. Dhamdhere. A new algorithm for composite hoisting and strength reduction. *International Journal of Computer Mathematics*, 27(1):1–14, 1989.

[121] Dhananjay M. Dhamdhere. Practical adaptation of the global optimization algorithm of Morel and Renvoise. *ACM Transactions on Programming Languages and Systems*, 13(2):291–294, April 1991.

[122] Michael K. Donegan, Robert E. Noonan, and Stefan Feyock. A code generator generator language. *SIGPLAN Notices*, 14(8):58–64, August 1979. *Proceedings of the ACM SIGPLAN '79 Symposium on Compiler Construction.*

[123] Jack J. Dongarra, James R. Bunch, Cleve B. Moler, and G. W. Stewart. *LINPACK User's Guide. SIAM*, Philadelphia, PA, 1979.

[124] Karl-Heinz Drechsler and Manfred P. Stadel. A solution to a problem with Morel and Renvoise's "Global optimization by suppression of partial redundancies." *ACM Transactions on Programming Languages and Systems*, 10(4):635–640, October 1988.

[125] Karl-Heinz Drechsler and Manfred P. Stadel. A variation of Knoop, Rüthing, and Steffen's "Lazy code motion." *SIGPLAN Notices*, 28(5):29–38, May 1993.

[126] Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, February 1970.

[127] Kemal Ebcioglu and Toshio Nakatani. A new compilation technique for parallelizing loops with unpredictable branches on a VLIW architecture. *Selected Papers of the Second Workshop on Languages and Compilers for Parallel Computing*, Pitman Publishing, London, UK, pages 213–229, 1990.

[128] John R. Ellis. *Bulldog: A Compiler for* VLIW *Architectures*. The MIT Press, Cambridge, MA, 1986.

[129] Maryam Emami, Rakesh Ghiya, and Laurie J. Hendren. Context-sensitive interprocedural points-to analysis in the presence of function pointers. *SIGPLAN Notices*, 29(6):242–256, June 1994. *Proceedings of the ACM SIGPLAN '94 Conference on Programming Language Design and Implementation.*

[130] Jens Ernst, William S. Evans, Christopher W. Fraser, Steven Lucco, and Todd A. Proebsting. Code compression. *SIGPLAN Notices*, 32(5):358–365, May 1997. In *Proceedings of the ACM SIGPLAN '97 Conference on Programming Language Design and Implementation.*

[131] Andrei P. Ershov. On programming of arithmetic expressions. *Communications of the ACM*, 1(8):3–6, August 1958. (The figures appear in volume 1, number 9, page 16.)

[132] Andrei P. Ershov. Reduction of the problem of memory allocation in programming to the problem of coloring the vertices of graphs. *Soviet Mathematics*, 3:163–165, 1962. Originally published in *Doklady Akademii Nauk S.S.S.R.*, 142(4), 1962.

[133] Andrei P. Ershov. Alpha—An automatic programming system of high efficiency. *Journal of the ACM*, 13(1):17–24, January 1966.

[134] Janet Fabri. Automatic storage optimization. *SIGPLAN Notices*, 14(8):83–91, August 1979. *Proceedings of the ACM SIGPLAN '79 Symposium on Compiler Construction.*

[135] Rodney Farrow. Linguist-86: Yet another translator writing system based on attribute grammars. *SIGPLAN Notices*, 17(6):160–171, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction.*

[136] Rodney Farrow. Automatic generation of fixed-point-finding evaluators for circular, but well-defined, attribute grammars. *SIGPLAN Notices*, 21(7):85–98, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction.*

[137] Robert R. Fenichel and Jerome C. Yochelson. A *LISP* garbage-collector for virtual-memory computer systems. *Communications of the ACM*, 12(11):611–612, November 1969.

[138] Jeanne Ferrante, Karl J. Ottenstein, and Joe D. Warren. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems*, 9(3):319–349, July 1987.

[139] Charles N. Fischer and Richard J. LeBlanc, Jr. The implementation of run-time diagnostics in Pascal. *IEEE Transactions on Software Engineering*, SE-6(4):313–319, 1980.

[140] Charles N. Fischer and Richard J. LeBlanc, Jr. *Crafting a Compiler with C*. Benjamin/Cummings, Redwood City, CA, 1991.

[141] Joseph A. Fisher. Trace scheduling: A technique for global microcode compaction. *IEEE Transactions on Computers*, C-30(7):478–490, July 1981.

[142] Joseph A. Fisher, John R. Ellis, John C. Ruttenberg, and Alexandru Nicolau. Parallel processing: A smart compiler and a dumb machine. *SIGPLAN Notices*, 19(6):37–47, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*.

[143] Robert W. Floyd. An algorithm for coding efficient arithmetic expressions. *Communications of the ACM*, 4(1):42–51, January 1961.

[144] J. M. Foster. A syntax improving program. *Computer Journal*, 11(1): 31–34, May 1968.

[145] Christopher W. Fraser. Automatic inference of models for statistical code compression. *SIGPLAN Notices*, 34(5):242–246, May 1999. In *Proceedings of the ACM SIGPLAN '99 Conference on Programming Language Design and Implementation*.

[146] Christopher W. Fraser, David R. Hanson, and Todd A. Proebsting. Engineering a simple, efficient code generator generator. *ACM Letters on Programming Languages and Systems*, 1(3):213–226, September 1992.

[147] Christopher W. Fraser and Robert R. Henry. Hard-coding bottom-up code generation tables to save time and space. *Software—Practice and Experience*, 21(1):1–12, January 1991.

[148] Christopher W. Fraser, Eugene W. Myers, and Alan L. Wendt. Analyzing and compressing assembly code. *SIGPLAN Notices*, 19(6):117–121, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*.

[149] Christopher W. Fraser and Alan L. Wendt. Integrating code generation and optimization. *SIGPLAN Notices*, 21(7):242–248, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction*.

[150] Christopher W. Fraser and Alan L. Wendt. Automatic generation of fast optimizing code generators. *SIGPLAN Notices*, 23(7):79–84, July 1988. *Proceedings of the ACM SIGPLAN '88 Conference on Programming Language Design and Implementation*.

[151] Mahadevan Ganapathi and Charles N. Fischer. Description-driven code generation using attribute grammars. In *Conference Record of the Ninth Annual ACM Symposium on Principles of Programming Languages*, pages 108–119, Albuquerque, NM, January 1982.

[152] Harald Ganzinger, Robert Giegerich, Ulrich Möncke, and Reinhard Wilhelm. A truly generative semantics-directed compiler generator. *SIGPLAN Notices*, 17(6):172–184, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction*.

[153] Lal George and Andrew W. Appel. Iterated register coalescing. In *Conference Record of the Twenty-Third ACM Symposium on Principles of Programming Languages*, pages 208–218, St. Petersburg Beach, FL, January 1996.

[154] Phillip B. Gibbons and Steven S. Muchnick. Efficient instruction scheduling for a pipelined architecture. *SIGPLAN Notices*, 21(7):11–16, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction.*

[155] R. Steven Glanville and Susan L. Graham. A new method for compiler code generation. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 231–240, Tucson, AZ, January 1978.

[156] Nikolas Gloy and Michael D. Smith. Procedure placement using temporal-ordering information. *ACM Transactions on Programming Languages and Systems*, 21(5):997–1027, September 1999.

[157] Adele Goldberg and David Robson. *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley, Reading, MA, 1983.

[158] James R. Goodman and Wei-Chung Hsu. Code scheduling and register allocation in large basic blocks. *Proceedings of the Second International Conference on Supercomputing*, pages 442–452, July 1988.

[159] Eiichi Goto. Monocopy and associative operations in extended Lisp. Technical Report 74-03, University of Tokyo, Tokyo, Japan, May 1974.

[160] Susan L. Graham. Table-driven code generation. *IEEE Computer*, 13(8):25–34, August 1980.

[161] Susan L. Graham, Michael A. Harrison, and Walter L. Ruzzo. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems*, 2(3):415–462, July 1980.

[162] Susan L. Graham, Robert R. Henry, and Robert A. Schulman. An experiment in table driven code generation. *SIGPLAN Notices*, 17(6):32–43, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction.*

[163] Susan L. Graham and Mark Wegman. A fast and usually linear algorithm for global flow analysis. In *Conference Record of the Second ACM Symposium on Principles of Programming Languages*, pages 22–34, Palo Alto, CA, January 1975.

[164] Susan L. Graham and Mark Wegman. A fast and usually linear algorithm for global flow analysis. *Journal of the ACM*, 23(1):172–202, 1976.

[165] Torbjörn Granlund and Richard Kenner. Eliminating branches using a superoptimizer and the *GNU* C compiler. *SIGPLAN Notices*, 27(7):341–352, July 1992. *Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation.*

[166] David Gries. *Compiler Construction for Digital Computers*. John Wiley and Sons, New York, NY, 1971.

[167] Rajiv Gupta and Mary Lou Soffa. Region scheduling: An approach for detecting and redistributing parallelism. *IEEE Transactions on Software Engineering*, 16(4):421–431, April 1990.

[168] Rajiv Gupta, Mary Lou Soffa, and Tim Steele. Register allocation via clique separators. *SIGPLAN Notices*, 24(7):264–274, July 1989. *Proceedings of the ACM SIGPLAN '89 Conference on Programming Language Design and Implementation.*

[169] Max Hailperin. Cost-optimal code motion. *ACM Transactions on Programming Languages and Systems*, 20(6):1297–1322, November 1998.

[170] Mary W. Hall and Ken Kennedy. Efficient call graph analysis. *ACM Letters on Programming Languages and Systems*, 1(3):227–242, September 1992.

[171] David R. Hanson. Fast allocation and deallocation of memory based on object lifetimes. *Software—Practice and Experience*, 20(1):5–12, January 1990.

[172] Dov Harel. A linear time algorithm for finding dominators in flow graphs and related problems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing* (STOC), pages 185–194, May 1985.

[173] William H. Harrison. A class of register allocation algorithms. Technical Report RC-5342, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1975.

[174] William H. Harrison. A new strategy for code generation—The general purpose optimizing compiler. *IEEE Transactions on Software Engineering*, SE-5(4):367–373, July 1979.

[175] Philip J. Hatcher and Thomas W. Christopher. High-quality code generation via bottom-up tree pattern matching. In *Conference Record of the Thirteenth Annual ACM Symposium on Principles of Programming Languages*, pages 119–130, St. Petersburg Beach, FL, January 1986.

[176] Matthew S. Hecht and Jeffrey D. Ullman. Characterizations of reducible flow graphs. *Journal of the ACM*, 21(3):367–375, July 1974.

[177] Matthew S. Hecht and Jeffrey D. Ullman. A simple algorithm for global data flow analysis problems. *SIAM Journal on Computing*, 4(4):519–532, 1975.

[178] J. Heller. Sequencing aspects of multiprogramming. *Journal of the ACM*, 8(3):426–439, July 1961.

[179] John L. Hennessy and Thomas Gross. Postpass code optimization of pipeline constraints. *ACM Transactions on Programming Languages and Systems*, 5(3):422–448, July 1983.

[180] Michael Hind, Michael Burke, Paul Carini, and Jong-Deok Choi. Interprocedural pointer alias analysis. *ACM Transactions on Programming Languages and Systems*, 21(4):848–894, July 1999.

[181] Michael Hind and Anthony Pioli. Which pointer analysis should I use? *ACM SIGSOFT Software Engineering Notes*, 25(5):113–123, September 2000. In *Proceedings of the International Symposium on Software Testing and Analysis*.

[182] Christoph M. Hoffmann and Michael J. O'Donnell. Pattern matching in trees. *Journal of the ACM*, 29(1):68–95, January 1982.

[183] John E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations: Proceedings*, pages 189–196, Academic Press, New York, NY, 1971.

[184] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.

[185] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press, Inc., Potomac, MD, 1978.

[186] Lawrence P. Horwitz, Richard M. Karp, Raymond E. Miller, and Shmuel Winograd. Index register allocation. *Journal of the ACM*, 13(1):43–61, January 1966.

[187] Susan Horwitz, Phil Pfeiffer, and Thomas Reps. Dependence analysis for pointer variables. *SIGPLAN Notices*, 24(7):28–40, July 1989. *Proceedings of the ACM SIGPLAN '89 Conference on Programming Language Design and Implementation*.

[188] Susan Horwitz and Tim Teitelbaum. Generating editing environments based on relations and attributes. *ACM Transactions on Programming Languages and Systems*, 8(4):577–608, October 1986.

[189] Brett L. Huber. Path-selection heuristics for dominator-path scheduling. Master's thesis, Computer Science Department, Michigan Technological University, Houghton, MI, October 1995.

[190] Wen-Mei W. Hwu, Scott A. Mahlke, William Y. Chen, Pohua P. Chang, Nancy J. Warter, Roger A. Bringmann, Roland G. Ouellette, Richard E. Hank, Tokuzo Kiyohara, Grant E. Haab, John G. Holm, and Daniel M. Lavery. The superblock: An effective technique for VLIW and superscalar compilation. *Journal of Supercomputing—Special Issue on Instruction Level Parallelism*, 7(1–2): 229–248, Kluwer Academic Publishers, Hingham, MA, May 1993.

[191] Edgar T. Irons. A syntax directed compiler for Algol 60. *Communications of the ACM*, 4(1):51–55, January 1961.

[192] J. R. Isaac and Dhananjay M. Dhamdhere. A composite algorithm for strength reduction and code movement optimization. *International Journal of Computer and Information Sciences*, 9(3):243–273, June 1980.

[193] Mehdi Jazayeri and Kenneth G. Walter. Alternating semantic evaluator. In *Proceedings of the 1975 Annual Conference of the ACM*, pages 230–234, 1975.

[194] Mark Scott Johnson and Terrence C. Miller. Effectiveness of a machine-level, global optimizer. *SIGPLAN Notices*, 21(7):99–108, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction*.

[195] Stephen C. Johnson. Yacc—Yet another compiler-compiler. Technical Report 32 (Computing Science), AT&T Bell Laboratories, Murray Hill, NJ, 1975.

[196] Stephen C. Johnson. A tour through the portable C compiler. In *Unix Programmer's Manual, 7th Edition*, volume 2b. AT&T Bell Laboratories, Murray Hill, NJ, January 1979.

[197] Walter L. Johnson, James H. Porter, Stephanie I. Ackley, and Douglas T. Ross. Automatic generation of efficient lexical processors using finite state techniques. *Communications of the ACM*, 11(12):805–813, December 1968.

[198] S. M. Joshi and Dhananjay M. Dhamdhere. A composite hoisting-strength reduction transformation for global program optimization. *International Journal of Computer Mathematics*, 11(1):21–44 (part I); 11(2): 111–126 (part II), 1982.

[199] John B. Kam and Jeffrey D. Ullman. Global data flow analysis and iterative algorithms. *Journal of the ACM*, 23(1):158–171, January 1976.

[200] John B. Kam and Jeffrey D. Ullman. Monotone data flow analysis frameworks. *Acta Informatica*, 7:305–317, 1977.

[201] Tadao Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. Scientific Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, 1965.

[202] Ken Kennedy. *Global Flow Analysis and Register Allocation for Simple Code Structures*. PhD thesis, Courant Institute, New York University, October 1971.

[203] Ken Kennedy. Global dead computation elimination. SETL Newsletter 111, Courant Institute of Mathematical Sciences, New York University, New York, NY, August 1973.

[204] Ken Kennedy. Reduction in strength using hashed temporaries. SETL Newsletter 102, Courant Institute of Mathematical Sciences, New York University, New York, NY, March 1973.

[205] Ken Kennedy. Node listings applied to data flow analysis. In *Conference Record of the Second ACM Symposium on Principles of Programming Languages*, pages 10–21, Palo Alto, CA, January 1975.

[206] Ken Kennedy. Use-definition chains with applications. *Computer Languages*, 3(3):163–179, 1978.

[207] Ken Kennedy. A survey of data flow analysis techniques. In Neil D. Jones and Steven S. Muchnick, editors, *Program Flow Analysis: Theory and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[208] Ken Kennedy and Linda Zucconi. Applications of graph grammar for program control flow analysis. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages*, pages 72–85, Los Angeles, CA, January 1977.

[209] Robert Kennedy, Fred C. Chow, Peter Dahl, Shin-Ming Liu, Raymond Lo, and Mark Streich. Strength reduction via SSAPRE. In *Proceedings of the Seventh International Conference on Compiler Construction. Lecture Notes in Computer Science 1383*, pages 144–158, Springer-Verlag, Heidelberg, Germany, March 1998.

[210] Daniel R. Kerns and Susan J. Eggers. Balanced scheduling: Instruction scheduling when memory latency is uncertain. *SIGPLAN Notices*, 28(6):278–289, June 1993. *Proceedings of the ACM SIGPLAN '93 Conference on Programming Language Design and Implementation.*

[211] Robert R. Kessler. Peep—An architectural description driven peephole optimizer. *SIGPLAN Notices*, 19(6):106–110, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction.*

[212] Gary A. Kildall. A unified approach to global program optimization. In *Conference Record of the ACM Symposium on Principles of Programming Languages*, pages 194–206, Boston, MA, October 1973.

[213] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Claude E. Shannon and John McCarthy, editors, *Automata Studies. Annals of Mathematics Studies*, 34:3–41. Princeton University Press, Princeton, NJ, 1956.

[214] Kath Knobe and Andrew Meltzer. Control tree based register allocation. Technical report, COMPASS, 1990.

[215] Jens Knoop, Oliver Rüthing, and Bernhard Steffen. Lazy code motion. *SIGPLAN Notices*, 27(7):224–234, July 1992. *Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation.*

[216] Jens Knoop, Oliver Rüthing, and Bernhard Steffen. Lazy strength reduction. *International Journal of Programming Languages*, 1(1):71–91, March 1993.

[217] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, December 1965.

[218] Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):127–145, 1968.

[219] Donald E. Knuth. Semantics of context-free languages: Correction. *Mathematical Systems Theory*, 5(1):95–96, 1971.

[220] Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 1973.

[221] Donald E. Knuth. A history of writing compilers. *Computers and Automation*, 11(12): 8–18, December 1962. Reprinted in *Compiler Techniques*, Bary W. Pollack, editor, pages 38–56, Auerbach, Princeton, NJ, 1972.

[222] Dexter C. Kozen. *Automata and Computability*. Springer-Verlag, New York, NY, 1997.

[223] Glenn Krasner, editor. *Smalltalk-80: Bits of History, Words of Advice*. Addison-Wesley, Reading, MA, August 1983.

[224] Sanjay M. Krishnamurthy. A brief survey of papers on scheduling for pipelined processors. *SIGPLAN Notices*, 25(7):97–106, July 1990.

[225] Steven M. Kurlander and Charles N. Fischer. Zero-cost range splitting. *SIGPLAN Notices*, 29(6):257–265, June 1994. *Proceedings of the ACM SIGPLAN '94 Conference on Programming Language Design and Implementation*.

[226] Monica Lam. Software pipelining: An effective scheduling technique for VLIW machines. *SIGPLAN Notices*, 23(7):318–328, July 1988. *Proceedings of the ACM SIGPLAN '88 Conference on Programming Language Design and Implementation*.

[227] David Alex Lamb. Construction of a peephole optimizer. *Software—Practice and Experience*, 11(6):639–647, June 1981.

[228] William Landi and Barbara G. Ryder. Pointer-induced aliasing: A problem taxonomy. In *Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages*, pages 93–103, Orlando, FL, January 1991.

[229] David Landskov, Scott Davidson, Bruce Shriver, and Patrick W. Mallett. Local microcode compaction techniques. *ACM Computing Surveys*, 12(3):261–294, September 1980.

[230] Rudolf Landwehr, Hans-Stephan Jansohn, and Gerhard Goos. Experience with an automatic code generator generator. *SIGPLAN Notices*, 17(6):56–66, June 1982. *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction*.

[231] James R. Larus and Paul N. Hilfinger. Register allocation in the SPUR Lisp compiler. *SIGPLAN Notices*, 21(7):255–263, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction*.

[232] S. S. Lavrov. Store economy in closed operator schemes. *Journal of Computational Mathematics and Mathematical Physics*, 1(4):687–701, 1961. English translation in *U.S.S.R. Computational Mathematics and Mathematical Physics* 3:810-828, 1962.

[233] Charles Lefurgy, Peter Bird, I-Cheng Chen, and Trevor Mudge. Improving code density using compression techniques. In *Proceedings of the Thirtieth International Symposium on Microarchitecture*, pages 194–203, IEEE Computer Society Press , Los Alamitos, CA, December 1997.

[234] Charles Lefurgy, Eva Piccininni, and Trevor Mudge. Reducing code size with run-time decompression. In *Proceedings of the Sixth International Symposium on High-Performance Computer Architecture*, pages 218–227, IEEE Computer Society Press, Los Alamitos, CA, January 2000.

[235] Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Transactions on Programming Languages and Systems*, 1(1):121–141, July 1979.

[236] Philip M. Lewis and Richard E. Stearns. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488, July 1968.

[237] Vincenzo Liberatore, Martin Farach-Colton, and Ulrich Kremer. Evaluation of algorithms for local register allocation. In *Eighth International Conference on Compiler Construction (CC 99). Lecture Notes in Computer Science 1575*, pages 137–152, Springer-Verlag, Heidelberg, Germany, 1999.

[238] Henry Lieberman and Carl Hewitt. A real-time garbage collector based on the lifetimes of objects. *Communications of the ACM*, 26(6):419–429, June 1983.

[239] Barbara Liskov, Russell R. Atkinson, Toby Bloom, J. Eliot B. Moss, Craig Schaffert, Robert Scheifler, and Alan Snyder. CLU *Reference Manual. Lecture Notes in Computer Science 114*, Springer-Verlag, Heidelberg, Germany, 1981.

[240] Jack L. Lo and Susan J. Eggers. Improving balanced scheduling with compiler optimizations that increase instruction-level parallelism. *SIGPLAN Notices*, 30(6):151–162, June 1995. *Proceedings of the ACM SIGPLAN '95 Conference on Programming Language Design and Implementation.*

[241] Raymond Lo, Fred Chow, Robert Kennedy, Shin-Ming Liu, and Peng Tu. Register promotion by sparse partial redundancy elimination of loads and stores. *SIGPLAN Notices*, 33(5):26–37, May 1998. *Proceedings of the ACM SIGPLAN '98 Conference on Programming Language Design and Implementation.*

[242] P. Geoffrey Lowney, Stefan M. Freudenberger, T. J. Karzes, W. D. Lichtenstein, Robert P. Nix, John S. O'Donnell, and John. C. Ruttenberg. The Multiflow trace scheduling compiler. *The Journal of Supercomputing—Special Issue*, 7(1–2):51–142, March 1993.

[243] Edward S. Lowry and C. W. Medlock. Object code optimization. *Communications of the ACM*, 12(1):13–22, January 1969.

[244] John Lu and Keith D. Cooper. Register promotion in C programs. *SIGPLAN Notices*, 32(5):308–319, May 1997. *Proceedings of the ACM SIGPLAN '97 Conference on Programming Language Design and Implementation.*

[245] John Lu and Rob Shillner. Clean: Removing useless control flow. Unpublished. Department of Computer Science, Rice University, Houston, TX, June 1994.

[246] Peter Lucas. Die strukturanalyse von formelubersetzern. *Elektronische Rechenanlagen*, 3:159–167, 1961.

[247] Guei-Yuan Lueh, Thomas Gross, and Ali-Reza Adl-Tabatabai. Global register allocation based on graph fusion. In *Proceedings of the Ninth International Workshop on Languages and Compilers for Parallel Computing (LCPC '96). Lecture Notes in Computer Science 1239*, pages 246–265, Springer-Verlag, Heidelberg, Germany, 1997.

[248] Bohdan S. Majewski, Nicholas C. Wormald, George Havas, and Zbigniew J. Czech. A family of perfect hashing methods. *The Computer Journal*, 39(6):547–554, 1996.

[249] Brian L. Marks. Compilation to compact code. *IBM Journal of Research and Development*, 24(6):684–691, November 1980.

[250] Peter W. Markstein, Victoria Markstein, and F. Kenneth Zadeck. Reassociation and strength reduction. Unpublished book chapter, July 1994.

[251] Henry Massalin. Superoptimizer—A look at the smallest program. *SIGPLAN Notices*, 22(10):122–126, October 1987. *Proceedings of the Second International Conference on Architectural Support for Programming Languages and Operating Systems.*

[252] John McCarthy. Lisp—Notes on its past and future. In *Proceedings of the 1980 ACM Conference on LISP and Functional Programming*, pages v–viii, Stanford University, Stanford, CA, 1980.

[253] William M. McKeeman. Peephole optimization. *Communications of the ACM*, 8(7):443–444, July 1965.

[254] Kathryn S. McKinley, Steve Carr, and Chau-Wen Tseng. Improving data locality with loop transformations. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 18(4):424–453, July 1996.

[255] Kathryn S. McKinley and Olivier Temam. A quantitative analysis of loop nest locality. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS-7), pages 94–104, Cambridge, MA, September 1996.

[256] Robert McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, EC-9(1):39–47, March 1960.

[257] Terrence C. Miller. *Tentative Compilation: A design for an APL compiler*. PhD thesis, Yale University, New Haven, CT, May 1978. See also paper of same title in *Proceedings of the International Conference on APL: Part 1*, pages 88–95, New York, NY, 1979.

[258] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML—Revised*. *MIT* Press, Cambridge, MA, 1997.

[259] Etienne Morel and Claude Renvoise. Global optimization by suppression of partial redundancies. *Communications of the ACM*, 22(2):96–103, February 1979.

[260] Robert Morgan. *Building an Optimizing Compiler*. Digital Press (an imprint of Butterworth–Heineman), Boston, MA, February 1998.

[261] Rajeev Motwani, Krishna V. Palem, Vivek Sarkar, and Salem Reyen. Combining register allocation and instruction scheduling. Technical Report 698, Courant Institute of Mathematical Sciences, New York University, New York, NY, July 1995.

[262] Steven S. Muchnick. *Advanced Compiler Design & Implementation*. Morgan Kaufmann, San Francisco, CA, August 1997.

[263] Frank Mueller and David B. Whalley. Avoiding unconditional jumps by code replication. *SIGPLAN Notices*, 27(7):322–330, July 1992. *Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation*.

[264] Thomas P. Murtagh. An improved storage management scheme for block structured languages. *ACM Transactions on Programming Languages and Systems*, 13(3):372–398, July 1991.

[265] Peter Naur (editor), J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Revised report on the algorithmic language Algol 60. *Communications of the ACM*, 6(1):1–17, January 1963.

[266] Brian R. Nickerson. Graph coloring register allocation for processors with multi-register operands. *SIGPLAN Notices*, 25(6):40–52, June 1990. *Proceedings of the ACM SIGPLAN '90 Conference on Programming Language Design and Implementation*.

[267] Cindy Norris and Lori L. Pollock. A scheduler-sensitive global register allocator. In *Proceedings of Supercomputing '93*, pages 804–813, Portland, OR, November 1993.

[268] Cindy Norris and Lori L. Pollock. An experimental study of several cooperative register allocation and instruction scheduling strategies. In *Proceedings of the Twenty-Eighth Annual International Symposium on Microarchitecture*, pages 169–179, IEEE Computer Society Press, Los Alamitos, CA, December 1995.

[269] Kristen Nygaard and Ole-Johan Dahl. The development of the *simula* languages. In *Proceedings of the First ACM SIGPLAN Conference on the History of Programming Languages*, pages 245–272, ACM Press, New York, NY, January 1978.

[270] Jinpyo Park and Soo-Mook Moon. Optimistic register coalescing. In *Proceedings of the 1998 International Conference on Parallel Architecture and Compilation Techniques* (PACT), pages 196–204, October 1998.

[271] Eduardo Pelegrí-Llopart and Susan L. Graham. Optimal code generation for expression trees: An application of BURS theory. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages*, pages 294–308, San Diego, CA, January 1988.

[272] Thomas J. Pennello. Very fast LR parsing. *SIGPLAN Notices*, 21(7):145–151, July 1986. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction.*

[273] Karl Pettis and Robert C. Hansen. Profile guided code positioning. *SIGPLAN Notices*, 25(6):16–27, June 1990. *Proceedings of the ACM SIGPLAN '90 Conference on Programming Language Design and Implementation.*

[274] Shlomit S. Pinter. Register allocation with instruction scheduling: A new approach. *SIGPLAN Notices*, 28(6):248–257, June 1993. *Proceedings of the ACM SIGPLAN '93 Conference on Programming Language Design and Implementation.*

[275] Gordon D. Plotkin. Call-by-name, call-by-value, and the λ-calculus. *Theoretical Computer Science*, 1(2):125–159, December 1975.

[276] Todd A. Proebsting. Simple and efficient BURS table generation. *SIGPLAN Notices*, 27(7):331–340, July 1992. *Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation.*

[277] Todd A. Proebsting. Optimizing an ANSI C interpreter with superoperators. In *Conference Record of the Twenty-Second ACM Symposium on Principles of Programming Languages*, pages 322–332, San Francisco, CA, January 1995.

[278] Todd A. Proebsting and Charles N. Fischer. Linear-time, optimal code scheduling for delayed-load architectures. *SIGPLAN Notices*, 26(6): 256–267, June 1991. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation.*

[279] Todd A. Proebsting and Charles N. Fischer. Probabilistic register allocation. *SIGPLAN Notices*, 27(7):300–310, July 1992. *Proceedings of the ACM SIGPLAN '92 Conference on Programming Language Design and Implementation.*

[280] Reese T. Prosser. Applications of boolean matrices to the analysis of flow diagrams. In *Proceedings of the Eastern Joint Computer Conference*, pages 133–138, Institute of Radio Engineers, New York, NY, December 1959.

[281] Paul W. Purdom, Jr. and Edward F. Moore. Immediate predominators in a directed graph [H]. *Communications of the ACM*, 15(8):777–778, August 1972.

[282] Brian Randell and L. J. Russell. *Algol 60 Programs Implementation; The Translation and Use of Algol 60 Programs on a Computer*. Academic Press, London, England, 1964.

[283] Bob R. Rau and C. D. Glaeser. Some scheduling techniques and an easily schedulable horizontal architecture for high performance scientific computing. In *Proceedings of the Fourteenth Annual Microprogramming Workshop on Microprogramming*, pages 183–198, December 1981.

[284] John H. Reif. Symbolic programming analysis in almost linear time. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 76–83, Tucson, AZ, January 1978.

[285] John H. Reif and Harry R. Lewis. Symbolic evaluation and the global value graph. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages*, pages 104–118, Los Angeles, CA, January 1977.

[286] Thomas Reps. Optimal-time incremental semantic analysis for syntax-directed editors. In *Conference Record of the Ninth Annual ACM Symposium on Principles of Programming Languages*, pages 169–176, Albuquerque, NM, January 1982.

[287] Thomas Reps and Bowen Alpern. Interactive proof checking. In *Conference Record of the Eleventh Annual ACM Symposium on Principles of Programming Languages*, pages 36–45, Salt Lake City, UT, January 1984.

[288] Thomas Reps and Tim Teitelbaum. *The Synthesizer Generator: A System for Constructing Language-Based Editors*. Springer-Verlag, New York, NY, 1988.

[289] Martin Richards. The portability of the BCPL compiler. *Software—Practice and Experience*, 1(2):135–146, April–June 1971.

[290] Steve Richardson and Mahadevan Ganapathi. Interprocedural analysis versus procedure integration. *Information Processing Letters*, 32(3): 137–142, August 1989.

[291] Ronald Rivest. On self-organizing sequential search heuristics. *Communications of the ACM*, 19(2):63–67, February 1976.

[292] Anne Rogers and Kai Li. Software support for speculative loads. *SIGPLAN Notices*, 27(9):38–50, September 1992. *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*.

[293] Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck. Global value numbers and redundant computations. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages*, pages 12–27, San Diego, CA, January 1988.

[294] Daniel J. Rosenkrantz and Richard Edwin Stearns. Properties of deterministic top-down grammars. *Information and Control*, 17(3):226–256, October 1970.

[295] Barbara G. Ryder. Constructing the call graph of a program. *IEEE Transactions on Software Engineering*, SE-5(3):216–226, May 1979.

[296] A. V. S. Sastry and Roy D. C. Ju. A new algorithm for scalar register promotion based on SSA form. *SIGPLAN Notices*, 33(5):15–25, May 1998. *Proceedings of the ACM SIGPLAN '98 Conference on Programming Language Design and Implementation*.

[297] Randolph G. Scarborough and Harwood G. Kolsky. Improved optimization of FORTRAN object programs. *IBM Journal of Research and Development*, 24(6):660–676, November 1980.

[298] Philip J. Schielke. *Stochastic Instruction Scheduling*. PhD thesis, Rice University, Department of Computer Science, Houston, TX, May 2000. (Technical Report TR00-370, Computer Science Department, Rice University, 2000.)

[299] Herb Schorr and William M. Waite. An efficient machine-independent procedure for garbage collection in various list structures. *Communications of the ACM*, 10(8):501–506, August 1967.

[300] Jacob T. Schwartz. On programming: An interim report on the SETL project. Installment II: The SETL language and examples of its use. Technical report, Courant Institute of Mathematical Sciences, New York University, New York, NY, October 1973.

[301] Ravi Sethi and Jeffrey D. Ullman. The generation of optimal code for arithmetic expressions. *Journal of the ACM*, 17(4):715–728, October 1970.

[302] Marc Shapiro and Susan Horwitz. Fast and accurate flow-insensitive points-to analysis. In *Conference Record of the Twenty-Fourth ACM Symposium on Principles of Programming Languages*, pages 1–14, Paris, France, January 1997.

[303] Robert M. Shapiro and Harry Saint. The representation of algorithms. Technical Report CA-7002-1432, Massachusetts Computer Associates, February 1970.

[304] Peter B. Sheridan. The arithmetic translator-compiler of the IBM FORTRAN automatic coding system. *Communications of the ACM*, 2(2):9–21, February 1959.

[305] Olin Shivers. Control flow analysis in Scheme. *SIGPLAN Notices*, 23(7):164–174, July 1988. *Proceedings of the ACM SIGPLAN '88 Conference on Programming Language Design and Implementation.*

[306] L. Taylor Simpson. *Value-Driven Redundancy Elimination*. PhD thesis, Rice University, Department of Computer Science, Houston, TX, 1996.(Technical Report TR 96–308, Computer Science Department, Rice University, 1996.)

[307] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Co., Boston, MA, December 1996.

[308] Richard L. Sites and Daniel R. Perkins. Universal P-code definition, version 0.2. Technical Report 78-CS-C29, Department of Applied Physics and Information Sciences, University of California at San Diego, San Diego, CA, January 1979.

[309] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2): 202–208, February 1985.

[310] Michael D. Smith, Mark Horowitz, and Monica S. Lam. Efficient superscalar performance through boosting. *SIGPLAN Notices*, 27(9): 248–259, September 1992. *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems.*

[311] Mark Smotherman, Sanjay M. Krishnamurthy, P. S. Aravind, and David Hunnicutt. Efficient DAG construction and heuristic calculation for instruction scheduling. In *Proceedings of the Twenty-Fourth Annual Workshop on Microarchitecture* (MICRO-24), pages 93–102, Albuquerque, NM, August 1991.

[312] Arthur Sorkin. Some comments on "A solution to a problem with Morel and Renvoise's 'Global optimization by suppression of partial redundancies.'" *ACM Transactions on Programming Languages and Systems*, 11(4):666–668, October 1989.

[313] Thomas C. Spillman. Exposing side-effects in a PL/1 optimizing compiler. In *Information Processing 71*, pages 376–381. North-Holland, Amsterdam, Netherlands, 1972. *Proceedings of IFIP Congress 71.*

[314] Guy Lewis Steele, Jr. Rabbit: A compiler for Scheme. Technical Report AI-TR-474, MIT Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, May 1978.

[315] Philip H. Sweany and Steven J. Beaty. Post-compaction register assignment in a retargetable compiler. In *Proceedings of the Twenty-Third Annual Workshop and Symposium on Microprogramming and Microarchitecture* (MICRO-23), pages 107–116, Orlando, FL, November 1990.

[316] Philip H. Sweany and Steven J. Beaty. Dominator-path scheduling—A global scheduling method. *ACM SIGMICRO Newsletter*, 23(1–2): 260–263, December 1992. *Proceedings of the Twenty-Fifth Annual International Symposium on Microarchitecture.*

[317] Robert Endre Tarjan. Testing flow graph reducibility. *Journal of Computer and System Sciences*, 9(3):355–365, December 1974.

[318] Robert Endre Tarjan. Fast algorithms for solving path problems. *Journal of the ACM*, 28(3):594–614, July 1981.

[319] Robert Endre Tarjan. A unified approach to path problems. *Journal of the ACM*, 28(3):577–593, July 1981.

[320] Robert Endre Tarjan and John H. Reif. Symbolic program analysis in almost-linear time. *SIAM Journal on Computing*, 11(1):81–93, February 1982.

[321] Ken Thompson. Programming Techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

[322] Steven W. K. Tjiang. TWIG reference manual. Technical Report CSTR 120, Computing Sciences, AT&T Bell Laboratories, Murray Hill, NJ, January 1986.

[323] Jeffrey D. Ullman. Fast algorithms for the elimination of common subexpressions. *Acta Informatica*, 2(3):191–213, 1973.

[324] David Ungar. Generation scavenging: A non-disruptive high performance storage reclamation algorithm. *ACM SIGSOFT Software Engineering Notes*, 9(3): 157–167, May 1984. *Proceedings of the First ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.*

[325] Victor Vyssotsky and Peter Wegner. A graph theoretical FORTRAN source language analyzer. Manuscript, AT&T Bell Laboratories, Murray Hill, NJ, 1963.

[326] William Waite and Gerhard Goos. *Compiler Construction*. Springer-Verlag, New York, NY, 1984.

[327] Scott Kipling Warren. *The Coroutine Model of Attribute Grammar Evaluation*. PhD thesis, Department of Mathematical Sciences, Rice University, Houston, TX, April 1976.

[328] Mark N. Wegman and F. Kenneth Zadeck. Constant propagation with conditional branches. In *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 291–299, New Orleans, LA, January 1985.

[329] Mark N. Wegman and F. Kenneth Zadeck. Constant propagation with conditional branches. *ACM Transactions on Programming Languages and Systems*, 13(2):181–210, April 1991.

[330] William E. Weihl. Interprocedural data flow analysis in the presence of pointers, procedure variables, and label variables. In *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages*, pages 83–94, Las Vegas, NV, January 1980.

[331] Clark Wiedmann. Steps toward an *APL* compiler. *ACM SIGAPL APL Quote Quad*, 9(4):321–328, June 1979. *Proceedings of the International Conference on APL.*

[332] Paul R. Wilson. Uniprocessor garbage collection techniques. In *Proceedings of the International Workshop on Memory Management. Lecture Notes in Computer Science 637*, pages 1–42, Springer-Verlag, Heidelberg, Germany, 1992.

[333] Robert P. Wilson and Monica S. Lam. Efficient context-sensitive pointer analysis for C programs. *SIGPLAN Notices*, 30(6):1–12, June 1995. *Proceedings of the ACM SIGPLAN '95 Conference on Programming Language Design and Implementation.*

[334] Michael E. Wolf and Monica S. Lam. A data locality optimizing algorithm. *SIGPLAN Notices*, 26(6):30–44, June 1991. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation.*

[335] Michael Wolfe. *High Performance Compilers for Parallel Computing*. Addison-Wesley, Redwood City, CA, 1996.

[336] D. Wood. The theory of left-factored languages, part 1. *The Computer Journal*, 12(4):349–356, November 1969.

[337] D. Wood. The theory of left-factored languages, part 2. *The Computer Journal*, 13(1):55–62, February 1970.

[338] D. Wood. A further note on top-down deterministic languages. *The Computer Journal*, 14(4):396–403, November 1971.

[339] William Wulf, Richard K. Johnsson, Charles B. Weinstock, Steven O. Hobbs, and Charles M. Geschke. *The Design of an Optimizing Compiler*. Programming Languages Series. American Elsevier Publishing Company, Inc., New York, NY, 1975.

[340] Cliff Young, David S. Johnson, David R. Karger, and Michael D. Smith. Near-optimal intraprocedural branch alignment. *SIGPLAN Notices*, 32(5):183–193, May 1997. *Proceedings of the ACM SIGPLAN '97 Conference on Programming Language Design and Implementation.*

[341] Daniel H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

[342] F. Kenneth Zadeck. Incremental data flow analysis in a structured program editor. *SIGPLAN Notices*, 19(6):132–143, June 1984. *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction.*