Papers on Compiler Optimizations: Analysis and Transformations (1952-1994)

Federico Bruzzone
Computer Science Department
Università degli Studi di Milano
federico.bruzzone@unimi.it
https://federicobruzzone.github.io/

30 September 2025

Title	Venue Type	Venue Name	Type	Year	Sources
The problem of simplifying truth functions [9]	T&F	AMM	J	1952	O
Minimization of Boolean functions [10]	Bell Labs	Bell System Tech. J.	J	1956	O
An algorithm for translating Boolean expressions [11]	ACM	JACM	J	1962	S
High speed compilation of efficient object code [12]	ACM	CACM	J	1965	D
Peephole optimization [13]	ACM	CACM	J	1965	E
Program optimization [14]	-	PP	C	1966	A, D, E, M, O, S
Index Register Allocation [15]	ACM	JACM	J	1966	E
Analysis of Programs for Parallel Processing [16]	IEEE	TC (TEC)	J	1966	O
Object code optimization [17]	ACM	CACM	J	1969	A, D, E
Local optimizations [18]	ACM	PLDI (SCO)	C	1970	E
Detection and parallel execution of independent instructions [19]	IEEE	TCO	J	1970	S
Global common subexpression elimination [20]	ACM	PLDI (SCO)	C	1970	D, E, M
The Generation of Optimal Code for Arithmetic Expressions [21]	ACM	JACM	J	1970	D, E, M
Control flow analysis [22]	ACM	PLDI (SCO)	C	1970	D, E
A Basis for Program Optimization [23]	IFIP	NH	C	1971	A, D, E, M, O, S
A catalogue of optimizing transformations [?]	-	PH	C	1972	A, D, E, M, O, S
Flow graph reducibility [24]	ACM	STOC	C	1972	D
Use-definition chains with applications [25]	Elsevier	COLA (COMLAN)	C	1972	E
A global flow analysis algorithm [26]	T&F	JCM	J	1972	E
Safety of code motion [27]	T&F	JCM	J	1972	O
On the Number of Operations Simultaneously Executable in Fortran-Like Programs and Their Resulting Speedup [28]	IEEE	TCO	J	1972	D, O
Testing flow graph reducibility [29]	ACM	STOC	C	1973	A
A unified approach to global program optimization [30]	ACM	POPL	C	1973	A, D, E, M, O, S
Fast algorithms for the elimination of common subexpressions [31]	Springer	Acta Inf.	J	1973	A, D, E, O
Register allocation via usage counts [32]	ACM	CACM	J	1974	A
Analysis of structured programs [33]	ACM	STOC	C	1974	D
Characterizations of Reducible Flow Graphs [34]	ACM	JACM	J	1974	D, E
The parallel execution of DO loops [35]	ACM	CACM	J	1974	D, O
Interprocedural Data Flow Analysis [36]	-	IFIP	J	1974	D, O

Continued on next page

Title	Venue Type	Venue Name	Type	Year	Sources
Time and parallel processor bounds for linear recurrence systems [37]	IEEE	TCO	J	1975	S
A simple algorithm for global data flow analysis problems [38]	SIAM	SICOMP	J	1975	A, E, O
A program data flow analysis procedure [39]	ACM	CACM	J	1976	A
Recursion analysis for compiler optimization [40]	ACM	CACM	J	1976	O
Optimal Code Generation for Expression Trees [41]	ACM	JACM	J	1976	A, D, E
A Fast and Usually Linear Algorithm for Global Flow Analysis [42]	ACM	JACM	J	1976	E, O
Code generation for expressions with common subexpressions [43]	ACM	POPL	C	1976	E, S
Global data flow analysis and iterative algorithms [44]	ACM	JACM	J	1976	E, O
On Live-Dead Analysis for Global Data Flow Problems [45]	ACM	JACM	J	1977	A
Symbolic evaluation and the global value graph [46]	ACM	POPL	C	1977	A
High-level data flow analysis [47]	ACM	CACM	J	1977	A
Abstract interpretation [48]	ACM	POPL	C	1977	D
An algorithm for reduction of operator strength [49]	ACM	CACM	J	1977	E
A transformation system for developing recursive programs [50]	ACM	JACM	J	1977	S
Arithmetic shifting considered harmful [51]	ACM	SIGPLAN Notices	J	1977	S
Monotone data flow analysis frameworks [52]	Springer	Acta Inf.	J	1977	A, D, E, O
Program Improvement by Source-to-Source Transformation [53]	ACM	CACM	J	1977	D, O
An analysis of inline substitution for a structured programming language [54]	ACM	CACM	J	1977	M, S
A new method for compiler code generation [55]	ACM	POPL	C	1978	D
A practical interprocedural data flow analysis algorithm [56]	ACM	CACM	J	1978	A, D, E, O
Data Flow Analysis for Procedural Languages [57]	ACM	JACM	J	1979	A
Constructing the Call Graph of a Program [58]	IEEE	TSE	J	1979	O
Data flow languages [59]	IEEE	MARK	W	1979	S
Time and parallel processor bounds for Fortran-like loops [60]	IEEE	TCO	J	1979	S
Unrolling loops in Fortran [61]	Wiley	SPE	J	1979	S
A fast algorithm for finding dominators in a flowgraph [62]	ACM	TOPLAS	J	1979	A, E, M, O
An efficient way to find the side effects of procedural calls and the aliases of variables [63]	ACM	POPL	C	1979	A, D, E, S
Global optimization by suppression of partial redundancies [64]	ACM	CAMC	J	1979	A, D, E, O, S
Predicting the effects of optimization on a procedure body [65]	ACM	PLDI (SCC)	C	1979	E, S
Structural analysis: A new approach to flow analysis in optimizing compilers [66]	Elsevier	COLA (COMLAN)	C	1980	A
The design and application of a retargetable peephole optimizer [67]	ACM	TOPLAS	J	1980	E
Data flow supercomputers [68]	IEEE	Computer	J	1980	S
High-speed multiprocessors and compilation techniques [69]	IEEE	TCO	J	1980	S
Interprocedural data flow analysis in the presence of pointers, procedure variables, and label variables [70]	ACM	POPL	C	1980	A, E, O
Deciding Linear Inequalities by Computing Loop Residues [71]	ACM	JACM	J	1981	D
A precise inter-procedural data flow algorithm [72]	ACM	POPL	C	1981	O, S
Register allocation via coloring [73]	Elsevier	COLA (COMLAN)	C	1981	A, D, E, O, S
Reduction of operator strength [74]	NJ	Program flow analysis	J	1981	A, E, S
Dependence graphs and compiler optimizatlons [75]	ACM	POPL	C	1981	O, S
On the performance enhancement of paging systems through program analysis and transformations [76]	IEEE	TCO	J	1981	D, S
Optimizing delayed branches [77]	ACM	MICRO	W	1982	D
A composite hoisting-strength reduction transformation for global program optimization part i [78]	T&F	JCM	J	1982	E
Optimization of range checking [79]	ACM	PLDI (SCC)	C	1982	E

Continued on next page

Title	Venue Type	Venue Name	Type	Year	Sources
Register allocation and spilling via graph coloring [80]	ACM	PLDI	C	1982	A, D, E, M, O, S
Experience with the SETL optimizer [81]	ACM	TOPLAS	J	1983	S
Postpass Code Optimization of Pipeline Constraints [82]	ACM	TOPLAS	J	1983	A, E
Conversion of control dependence to data dependence [83]	ACM	POPL	C	1983	O, S
Register allocation and exhaustive peephole optimization [84]	Wiley	SPE	J	1984	E
Automatic generation of peephole optimizations [85]	Springer	CC	C	1984	E
Analysis of interprocedural side effects in a parallel programming environment [86]	Springer	ICSP	C	1984	O
Polyvariant mixed computation for analyzer programs [87]	Springer	Acta Inf.	J	1984	O
Stream processing [88]	ACM	LFP	C	1984	O
A hierarchical basis for reordering transformations [89]	ACM	POPL	C	1984	O
Register allocation by priority-based coloring [90]	ACM	PLDI (SCC)	C	1984	A, E, O
Automatic loop interchange [91]	ACM	PLDI (SCC)	C	1984	O, S
Efficient computation of flow insensitive interprocedural summary information [92]	ACM	PLDI (SCC)	C	1984	A, O
On linearizing parallel code [93]	ACM	POPL	C	1985	O
Distributed execution of functional programs using serial combinators [94]	IEEE	TCO	J	1985	S
Strictness analysis-a practical approach [95]	Springer	FPCA	C	1985	S
A linear algorithm for finding dominators in flow graphs and related problems [96]	ACM	STOC	C	1985	E, M, O
Efficient instruction scheduling for a pipelined architecture [97]	ACM	PLDI (SCC)	C	1986	A
Efficient symbolic analysis of programs [98]	ACM	JCSS	J	1986	A
Graph-Based Algorithms for Boolean Function Manipulation [99]	IEEE	TC	J	1986	D
Loops skewing: The wavefront method revisited [100]	Springer	JPP	J	1986	O
Highly concurrent scalar processing [101]	ACM	CAN	J	1986	O
Multiplication by integer constants [102]	Wiley	SPE	J	1986	S
Global register allocation at link time [103]	ACM	PLDI	C	1986	A, E
Interprocedural constant propagation [104]	ACM	PLDI	C	1986	A, D, E, O, S
Interprocedural optimization: eliminating unnecessary recompilation [105]	ACM	CC	C	1986	A, E, O
Interprocedural dependence analysis and parallelization [106]	ACM	PLDI	C	1986	O, S
Effectiveness of a machine-level, global optimizer [107],	ACM	PLDI (SCC)	C	1986	A, E
Direct parallelization of call statements [108]	ACM	PLDI (SCC)	C	1986	O, S
Code motion of control structures in high-level languages [109]	ACM	POPL	C	1986	E, M
Automatic inference and fast interpretation of peephole optimization rules [110]	Wiley	SPE	J	1987	E
Compiler Algorithms for Synchronization [111]	IEEE	TCO	J	1987	O
Automatic decomposition of scientific programs for parallel execution [112]	ACM	POPL	C	1987	O
Guided Self-Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers [113]	IEEE	TCO	J	1987	S
Loop coalescing: A compiler transformation for parallel machines [114]	ACM	ICPP	C	1987	S
Strategies for cache and local memory management by global program transformation [115]	Springer	ICS	C	1987	S
The program dependence graph and its use in optimization [116]	ACM	TOPLAS	J	1987	A, E, M, O
Automatic translation of Fortran programs to vector form [117]	ACM	TOPLAS	J	1987	D, O, S
An efficient approach to data flow analysis in a multiple pass global optimizer [118]	ACM	PLDI	C	1988	A
A solution to a problem with Morel and Renvoise's "Global optimization by suppression of partial redundancies" [119]	ACM	TOPLAS	J	1988	E

Continued on next page

Title	Venue Type	Venue Name	Type	Year	Sources
Efficient computation of flow-insensitive interprocedural summary information—a correction [120]	ACM	SIGPLAN Notices	J	1988	O
Dependence of multi-dimensional array references [121]	ACM	ICS	C	1988	O
The importance of direct dependences for automatic parallelization [122]	ACM	ICS	C	1988	O
Introducing symbolic problem solving techniques in the dependence testing phases of a vectorizer [123]	ACM	ICS	C	1988	O
Generating sequential code from parallel code [124]	ACM	SC	C	1988	O
Advanced loop optimizations for parallel computers [125]	ACM	ICS	C	1988	S
An introduction to a formal theory of dependence analysis [126]	Springer	JSC	J	1988	S
Analysis of interprocedural side effects in a parallel programming environment [127]	ACM	ICS	C	1988	S
Array expansion [128]	ACM	ICS	C	1988	S
Loop quantization: A generalized loop unwinding technique [129]	ACM	JPDC	J	1988	S
Supernode partitioning [130]	ACM	PLDI	C	1988	S
A fast algorithm for code movement optimisation [131]	ACM	PLDI	C	1988	A, E
Compiling programs for distributed-memory multiprocessors [132]	Elsevier	JSC	J	1988	O, S
Optimal loop parallelization [133]	ACM	PLDI	C	1988	A, E, M, S
Software pipelining: an effective scheduling technique for VLIW machines [134]	ACM	PLDI	C	1988	D, E
The program summary graph and flow-sensitive interprocedual data flow analysis [135]	ACM	PLDI	C	1988	A, O
A framework for determining useful parallelism [136]	ACM	ICS	C	1988	O, S
An overview of the PTRAN analysis system for multiprocessing [137]	Elsevier	JPDC	J	1988	D, S
Efficient interprocedural analysis for program parallelization and restructuring [138]	ACM	PPoPP (PPEALS)	C	1988	O, S
Compiling issues for supercomputers [139]	ACM/IEE	SC	C	1988	O, S
Minimizing register usage penalty at procedure calls [140]	ACM	PLDI	C	1988	A, S
Estimating interlock and improving balance for pipelined architectures [141]	Elsevier	JPDC	J	1988	O, S
Global value numbers and redundant computations [142]	ACM	POPL	C	1988	E, O
Perfect pipelining: A new loop parallelization technique [143]	Springer	ESOP	C	1988	A, S
Code scheduling and register allocation in large basic blocks [144]	ACM	SC	C	1988	A, E
Interprocedural side-effect analysis in linear time [145]	ACM	PLDI	C	1988	A, E, O
Unified management of registers and cache using liveness and cache bypass [146]	ACM	PLDI	C	1989	A
A new algorithm for composite hoisting and strength reduction optimisation [147]	T&F	JCM	J	1989	E
Program optimization for instruction caches [148]	ACM	ASPLOS	C	1989	A
Dependence analysis for pointer variables [149]	ACM	PLDI	C	1989	E
The program dependence graph and vectorization [150]	ACM	POPL	C	1989	O
Achieving high instruction cache performance with an optimizing compiler [151]	ACM	ISCA	C	1989	S
Evaluating the performance of four snooping cache coherency protocols [152]	IEEE	ICSA	C	1989	S
Scans as primitive parallel operations [153]	IEEE	TCO	J	1989	S
Code generation using tree matching and dynamic programming [154]	ACM	TOPLAS	J	1989	A, D, E, M
Register allocation via clique separators [155]	ACM	PLDI	C	1989	A, E
Fast interprocedual alias analysis [156]	ACM	POPL	C	1989	A, E, O, S
Coloring heuristics for register allocation [157]	ACM	PLDI	C	1989	A, E, O
More iteration space tiling [158]	ACM/IEE	SC	C	1989	A, S
Data dependence analysis on multi-dimensional array references [159]	ACM	ICS	C	1989	O, S

Continued on next page

Title	Venue Type	Venue Name	Type	Year	Sources
Spill code minimization techniques for optimizing compliers [160]	ACM	PLDI	C	1989	A, E
Customization: Optimizing compiler technology for SELF, a dynamically-typed OOP language [161]	ACM	PLDI	C	1989	O, S
An efficient method of computing static single assignment form [162]	ACM	POPL	C	1989	A, O
An approach to ordering optimizing transformations [163]	ACM	PPoPP	C	1990	A
Register allocation across procedure and module boundaries [164]	ACM	PLDI	C	1990	A
Region Scheduling: An Approach for Detecting and Redistributing Parallelism [165]	IEEE	TSE	J	1990	E
Constructing the procedure call multigraph [166]	IEEE	TSE	J	1990	O
On the perfect accuracy of an approximate subscript analysis test [167]	ACM	CAN	J	1990	O
Structured dataflow analysis for arrays and its use in an optimizing compiler [168]	Wiley	Software: Practice and Experience	J	1990	O
Compilation of Haskell array comprehensions for scientific computing [169]	ACM	PLDI	C	1990	S
How to read floating point numbers accurately [170]	ACM	PLDI	C	1990	S
How to print floating-point numbers accurately [171]	ACM	PLDI	C	1990	S
Profile guided code positioning [172]	ACM	PLDI	C	1990	S
Updating distributed variables in local computations [173]	Wiley	SPE	J	1990	S
An interval-based approach to exhaustive and incremental interprocedural data-flow analysis [174]	ACM	TOPLAS	J	1990	E, O
The priority-based coloring approach to register allocation [175]	ACM	TOPLAS	J	1990	A, D, E, S
Improving register allocation for subscripted variables [176]	ACM	PLDI	C	1990	A, E, M, O, S
Analysis of pointers and structures [177]	ACM	PLDI	C	1990	A, E
Loop distribution with arbitrary control flow [178]	ACM	PLDI	C	1990	O, S
Graph coloring register allocation for processors with multi-register operands [179]	ACM	PLDI	C	1990	A, E
Register allocation via hierarchical graph coloring [180]	ACM	PLDI	C	1991	A
Circular scheduling: a new technique to perform software pipelining [181]	ACM	PLDI	C	1991	A
Efficient DAG construction and heuristic calculation for instruction scheduling [182]	ACM	MICRO	W	1991	E
A composite algorithm for strength reduction and code movement optimization [183]	Springer	ACIS	J	1991	E
Efficiently computing static single assignment form and the control dependence graph [184]	ACM	TOPLAS	J	1991	M
Software prefetching [185]	ACM	ASPLOS	C	1991	O
Compiling global name-space parallel loops for distributed execution [186]	IEEE	TPDS	J	1991	O
An implementation of interprocedural bounded regular section analysis [187]	ACM	TPDS	J	1991	O
Limits of instruction-level parallelism [188]	ACM	ASPLOS	C	1991	S
Uniform techniques for loop optimization [189]	ACM	ICS	C	1991	S
A data locality optimizing [190]	ACM	PLDI	C	1991	A, D, M, O
Constant propagation with conditional branches [191]	ACM	TOPLAS	J	1991	A, E, M, O, S
Efficient and exact data dependence analysis [192]	ACM	PLDI	C	1991	A, D, O
Efficiently computing static single assignment [193]	ACM	TOPLAS	J	1991	A, D, E, O
Global instruction scheduling for superscalar machines [194]	ACM	PLDI	C	1991	A, D, E
Practical adaption of the global optimization algorithm of Morel and Renvoise [195]	ACM	TOPLAS	J	1991	A, E

Continued on next page

Title	Venue Type	Venue Name	Type	Year	Sources
The cache performance and optimizations of blocked algorithms [196]	ACM	ASPLOS	C	1991	A, D
A loop transformation theory and an algorithm to maximize parallelism [197]	IEEE	TPDS	J	1991	O, S
Dataflow analysis of array and scalar references [198]	Springer	JPP	J	1991	A, S
Optimization of array accesses by collective loop transformations [199]	ACM	SC	C	1991	D, O
Interprocedural transformations for parallel code generation [200]	ACM/IEE	SC	C	1991	O, S
Practical dependence testing [201]	ACM	PLDI	C	1991	A, O
Procedure merging with instruction caches [202]	ACM	PLDI	C	1991	A, S
An experiment with inline substitution [203]	Wiley	SPE	J	1991	E, S
Integrating scalar optimization and parallelization [204]	Springer	LCPC	C	1992	A
Sharlit—a tool for building optimizers [205]	ACM	PLDI	C	1992	A
How to analyze large programs efficiently and informatively [206]	ACM	PLDI	C	1992	A
Compiler code transformations for superscalar-based high performance systems [207]	IEEE	SC	C	1992	A
Engineering a simple, efficient code-generator generator [208]	ACM	TOPLAS (LOPLAS)	J	1992	D
Some efficient solutions to the affine scheduling problem. I. One-dimensional time [209]	Elsevier	JPP	J	1992	D
Avoiding unconditional jumps by code replication [210]	ACM	PLDI	C	1992	E
Rematerialization [211]	ACM	PLDI	C	1992	E
Coloring register pairs [212]	ACM	TOPLAS (LOPLAS)	J	1992	E
Optimizing for parallelism and data locality [213]	ACM	SC	C	1992	O
Maximizing loop parallelism and improving data locality via loop fusion and distribution [214]	ACM	LCPC	W	1992	O
Beyond induction variables [215]	ACM	PLDI	C	1992	O
Efficient call graph analysis [216]	ACM	TOPLAS (LOPLAS)	J	1992	O
A safe approximate algorithm for interprocedural aliasing [217]	ACM	PLDI	C	1992	O
Relaxing SIMD control flow constraints using loop transformations [218]	ACM	PLDI	C	1992	S
Unexpected side effects of inline substitution: a case study [219]	ACM	TOPLAS (LOPLAS)	J	1992	S
Eliminating false positives using the omega test [220]	ACM	PLDI	C	1992	A, D, O
Lazy code motion [221]	ACM	PLDI	C	1992	A, D, E
Software support for speculative loads [222]	ACM	ASPLOS	C	1992	A, E
A practical algorithm for exact array dependence analysis [223]	ACM	CACM	J	1992	O, S
Array privatization for parallel execution of loops [224]	ACM	ICS	C	1992	O, S
The power test for data dependence [225]	IEEE	TCO	J	1992	A, S
Eliminating branches using a superoptimizer and the GNU C compiler [226]	ACM	PLDI	C	1992	E, S
Design and evaluation of a compiler algorithm for prefetching [227]	ACM	ASPLOS	C	1992	D, O
Dependence-based program analysis [228]	ACM	PLDI	C	1993	A
A variation of Knoop, Rüthing, and Steffen's lazy code motion [229]	ACM	PLDI	C	1993	E
Lazy strength reduction [230]	C&H	JPL	J	1993	A, E
Collective loop fusion for array contraction [231]	Springer	LCPC	C	1993	O
Interprocedural constant propagation: an empirical study [232]	ACM	TOPLAS (LOPLAS)	J	1993	O
Symbolic analysis: A basis for parallelization, optimization, and scheduling of programs [233]	Springer	LCPC	W	1993	O
Array-data flow analysis and its use in array privatization [234]	ACM	POPL	C	1993	S
Automatic array alignment in data-parallel programs [235]	ACM	POPL	C	1993	S

Continued on next page

Title	Venue Type	Venue Name	Type	Year	Sources
Global optimizations for parallelism and locality on scalable parallel machines [236]	ACM	PLDI	C	1993	S
Instruction-Level Parallel Processing: History, Overview, and Perspective [237]	Springer	JSC	J	1993	S
A practical system for intermodule code optimization at link-time [238]	C&H	JPL	J	1993	A
Loop-level parallelism in numeric and symbolic programs [239]	IEEE	TPDS	J	1993	S
Orchestrating interactions among parallel computations [240]	ACM	PLDI	C	1993	S
The superblock: an effective technique for VLIW and superscalar compilation [241]	Springer	JSC	J	1993	S
Optimizing array bound checks using flow analysis [242]	ACM	PLDI	C	1993	A, E
Register allocation with instruction scheduling [243]	ACM	PLDI	C	1993	A, E
A methodology for procedure cloning [244]	Elsevier	COLA (COMLAN)	J	1993	A, O, S
Efficient flow-sensitive interprocedural computation of pointer-induced aliases and side effects [245]	ACM	POPL	C	1993	E, O, S
Automatic array privatization [246]	Springer	LCPC	W	1993	O, S
Interprocedural modification side effect analysis with pointer aliasing [247]	ACM	PLDI	C	1993	M, S
Branch prediction for free [248]	ACM	PLDI	C	1993	A, M
Link-time optimization of address calculation on a 64-bit architecture [249]	ACM	PLDI	C	1994	A
Effective partial redundancy elimination [250]	ACM	PLDI	C	1994	A
Partial dead code elimination [251]	ACM	PLDI	C	1994	A
A general data dependence test for dynamic, pointer-based data structures [252]	ACM	PLDI	C	1994	A
Instruction scheduling over regions: A framework for scheduling across basic blocks [253]	Springer	CC	C	1994	A
Value dependence graphs: representation without taxation [254]	ACM	POPL	C	1994	A
Zero-cost range splitting [255]	ACM	PLDI	C	1994	E
Reducing branch costs via branch alignment [256]	ACM	OSR	J	1994	E
Improving the ratio of memory operations to floating-point operations in loops [257]	ACM	TOPLAS	J	1994	M
Optimizing multi-method dispatch using compressed dispatch tables [258]	ACM	OOPSLA	C	1994	M
Improving the accuracy of static branch prediction using branch correlation [259]	ACM	ASPLOS	C	1994	M
A compiler framework for restructuring data declarations to enhance cache and TLB effectiveness [260]	IEEE	CASCON	C	1994	S
False sharing and spatial locality in multiprocessor caches [261]	IEEE	TCO	J	1994	S
Reassociation and strength reduction [262]	ACM	SCO	C	1994	S
The alignment-distribution graph [263]	Springer	LCPC	W	1994	S
Improvements to graph coloring register allocation [264]	ACM	TOPLAS	J	1994	A, E, M
Context-sensitive interprocedural points-to analysis in the presence of function pointers [265]	ACM	PLDI	C	1994	D, E
Interprocedural may-alias analysis for pointers: beyond k-limiting [266]	ACM	PLDI	C	1994	A, E

Scalar replacement in the presence of conditional control flow [267]	Wiley	SPE	J	1994	E, O

Table 1: The papers from *Advanced Compiler Design and Implementation* [1] (A), *Compilers: Principles, Techniques, and Tools* [2] (D), *Engineering a Compiler* [3] (E), *Modern Compiler Implementation in C/Java/ML* [4–6] (M), *Optimizing Compilers for Modern Architectures: A Dependence-Based Approach* [7] (O), and the Bacon's survey [8] (S). In the **Type** column, C stands for conference, J for journal, and W for workshop.

References

[1] Steven Muchnick. *Advanced compiler design implementation*. Morgan kaufmann, 1997.

[2] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Boston, MA, USA, second edition, 2006.

[3] Keith D. Cooper and Linda Torczon. *Engineering a Compiler*. Morgan Kaufmann, November 2022.

[4] Andrew W Appel. Modern compiler design in c, 1998.

[5] W Appel Andrew and Palsberg Jens. Modern compiler implementation in java. In *ISBN 0–521–58388–8*. Cambridge University Press, 2002.

[6] Andrew W Appel. *Modern compiler implementation in ML*. Cambridge university press, 1998.

[7] Ken Kennedy and John R Allen. *Optimizing compilers for modern architectures: a dependence-based approach*. Morgan Kaufmann Publishers Inc., 2001.

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

[9] Willard V Quine. The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531, 1952.

[10] Edward J McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444, 1956.

[11] Bruce W Arden, Bernard A Galler, and Robert M Graham. An algorithm for translating boolean expressions. *Journal of the ACM (JACM)*, 9(2):222–239, 1962.

[12] C. W. Gear. High speed compilation of efficient object code. *Commun. ACM*, 8(8):483–488, August 1965.

[13] W. M. McKeeman. Peephole optimization. *Commun. ACM*, 8(7):443–444, July 1965.

[14] Frances E. Allen. Program optimization. In Mark Halpern and Christopher Shaw, editors, *Annual Review of Automatic Programming*, volume 5, pages 239–307. Pergamon Press, Elmsford, NY, 1966.

[15] L. P. Horwitz, R. M. Karp, R. E. Miller, and S. Winograd. Index register allocation. *J. ACM*, 13(1):43–61, January 1966.

[16] A. J. Bernstein. Analysis of programs for parallel processing. *IEEE Transactions on Electronic Computers*, EC-15(5):757–763, 1966.

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

[18] John T Bagwell Jr. Local optimizations. In *Proceedings of a symposium on Compiler optimization*, pages 52–66, 1970.

[19] Garold S Tjaden and Michael J Flynn. Detection and parallel execution of independent instructions. *IEEE Transactions on computers*, 19(10):889–895, 1970.

[20] John Cocke. Global common subexpression elimination. In *Proceedings of a Symposium on Compiler Optimization*, page 20–24, New York, NY, USA, 1970. Association for Computing Machinery.

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

[22] Frances E. Allen. Control flow analysis. *SIGPLAN Not.*, 5(7):1–19, July 1970.

[23] Frances E. Allen. A basis for program optimization. In Charles V. Freiman, John E. Griffith, and Jack L. Rosenfeld, editors, *Information Processing, Proceedings of IFIP Congress 1971, Volume 1 - Foundations and Systems, Ljubljana, Yugoslavia, August 23-28, 1971*, pages 385–390. North-Holland, 1971.

[24] Matthew S. Hecht and Jeffrey D. Ullman. Flow graph reducibility. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, page 238–250, New York, NY, USA, 1972. Association for Computing Machinery.

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

[26] Ken Kennedy. A global flow analysis algorithm. *International Journal of Computer Mathematics*, 3(1-4):5–15, 1972.

[27] Ken Kennedy. Safety of code motion. *International Journal of Computer Mathematics*, 3(1-4):117–130, 1972.

[28] D.J. Kuck, Y. Muraoka, and Shyh-Ching Chen. On the number of operations simultaneously executable in fortran-like programs and their resulting speedup. *IEEE Transactions on Computers*, C-21(12):1293–1310, 1972.

[29] Robert Tarjan. Testing flow graph reducibility. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, page 96–107, New York, NY, USA, 1973. Association for Computing Machinery.

[30] Gary A Kildall. A unified approach to global program optimization. In *Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 194–206, 1973.

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

[32] Robert A Freiburghouse. Register allocation via usage counts. *Communications of the ACM*, 17(11):638–642, 1974.

[33] S. Rao Kosaraju. Analysis of structured programs. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, page 240–252, New York, NY, USA, 1973. Association for Computing Machinery.

[34] M. S. Hecht and J. D. Ullman. Characterizations of reducible flow graphs. *J. ACM*, 21(3):367–375, July 1974.

[35] Leslie Lamport. The parallel execution of do loops. *Commun. ACM*, 17(2):83–93, February 1974.

[36] Frances E. Allen. Interprocedural data flow analysis. In Jack L. Rosenfeld, editor, *Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974*, pages 398–402. North-Holland, 1974.

[37] Shyh-Ching Chen and David J. Kuck. Time and parallel processor bounds for linear recurrence systems. *IEEE Transactions on Computers*, 100(7):701–717, 1975.

[38] 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.

[39] F. E. Allen and J. Cocke. A program data flow analysis procedure. *Commun. ACM*, 19(3):137, March 1976.

[40] Kenneth G Walter. Recursion analysis for compiler optimization. *Communications of the ACM*, 19(9):514–516, 1976.

[41] A. V. Aho and S. C. Johnson. Optimal code generation for expression trees. *J. ACM*, 23(3):488–501, July 1976.

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

[43] Alfred V Aho, Stephen C Johnson, and Jeffrey D Ullman. Code generation for expressions with common subexpressions. In *Proceedings of the 3rd ACM SIGACT-SIGPLAN symposium on Principles on programming languages*, pages 19–31, 1976.

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

[45] Lawrence T. Kou. On live-dead analysis for global data flow problems. *J. ACM*, 24(3):473–483, July 1977.

[46] John H. Reif and Harry R. Lewis. Symbolic evaluation and the global value graph. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, page 104–118, New York, NY, USA, 1977. Association for Computing Machinery.

[47] Barry K. Rosen. High-level data flow analysis. *Commun. ACM*, 20(10):712–724, October 1977.

[48] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, page 238–252, New York, NY, USA, 1977. Association for Computing Machinery.

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

[50] Rod M Burstall and John Darlington. A transformation system for developing recursive programs. *Journal of the ACM (JACM)*, 24(1):44–67, 1977.

[51] Guy L. Steele. Arithmetic shifting considered harmful. *SIGPLAN Not.*, 12(11):61–69, November 1977.

[52] John B Kam and Jeffrey D Ullman. Monotone data flow analysis frameworks. *Acta informatica*, 7(3):305–317, 1977.

[53] David B. Loveman. Program Improvement by Source-to-Source Transformation. *Journal of the ACM*, 24(1):121–145, January 1977.

[54] Robert W. Scheifler. An analysis of inline substitution for a structured programming language. *Commun. ACM*, 20(9):647–654, September 1977.

[55] R. Steven Glanville and Susan L. Graham. A new method for compiler code generation. In *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '78, page 231–254, New York, NY, USA, 1978. Association for Computing Machinery.

[56] Jeffrey M. Barth. A practical interprocedural data flow analysis algorithm. *Commun. ACM*, 21(9):724–736, September 1978.

[57] Barry K. Rosen. Data flow analysis for procedural languages. *J. ACM*, 26(2):322–344, April 1979.

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

[59] William B Ackerman. Data flow languages. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 1087–1095. IEEE, 1979.

[60] Shyh-Ching Chen et al. Time and parallel processor bounds for fortran-like loops. *IEEE Transactions on Computers*, 100(9):660–670, 1979.

[61] Jack J Dongarra and A_R Hinds. Unrolling loops in fortran. *Software: Practice and Experience*, 9(3):219–226, 1979.

[62] Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph.

ACM Trans. Program. Lang. Syst., 1(1):121–141, January 1979.

[63] John P Banning. An efficient way to find the side effects of procedure calls and the aliases of variables. In *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 29–41, 1979.

[64] E. Morel and C. Renvoise. Global optimization by suppression of partial redundancies. *Commun. ACM*, 22(2):96–103, February 1979.

[65] J Eugene Ball. Predicting the effects of optimization on a procedure body. *ACM SIGPLAN Notices*, 14(8):214–220, 1979.

[66] M. Sharir. Structural analysis: A new approach to flow analysis in optimizing compilers. *Comput. Lang.*, 5(3–4):141–153, January 1980.

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

[68] Jack B Dennis. Data flow supercomputers. *Computer*, 13(11):48–56, 1980.

[69] Padua, Kuck, and Lawrie. High-speed multiprocessors and compilation techniques. *IEEE Transactions on Computers*, 100(9):763–776, 1980.

[70] William E. Weihl. Interprocedural data flow analysis in the presence of pointers, procedure variables, and label variables. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, page 83–94, New York, NY, USA, 1980. Association for Computing Machinery.

[71] Robert Shostak. Deciding linear inequalities by computing loop residues. *J. ACM*, 28(4):769–779, October 1981.

[72] Eugene M. Myers. A precise inter-procedural data flow algorithm. In *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '81, page 219–230, New York, NY, USA, 1981. Association for Computing Machinery.

[73] 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, 1981.

[74] Frances E Allen, John Cocke, and Ken Kennedy. Reduction of operator strength. *Program Flow Analysis*, pages 79–101, 1981.

[75] D. J. Kuck, R. H. Kuhn, D. A. Padua, B. Leasure, and M. Wolfe. Dependence graphs and compiler optimizations. In *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '81, page 207–218, New York, NY, USA, 1981. Association for Computing Machinery.

[76] Kuck and Lawrie. On the performance enhancement of paging systems through program analysis and transformations. *IEEE Transactions on Computers*, 100(5):341–356, 1981.

[77] Thomas R. Gross and John L. Hennessy. Optimizing delayed branches. *SIGMICRO Newsl.*, 13(4):114–120, October 1982.

[78] SM Joshi and Dhananjay M Dhamdhere. A composite hoisting-strength reduction transformation for global program optimization part i. *International Journal of Computer Mathematics*, 11(1):21–41, 1982.

[79] Victoria Markstein, John Cocke, and Peter Markstein. Optimization of range checking. In *Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction*, SIGPLAN '82, page 114–119, New York, NY, USA, 1982. Association for Computing Machinery.

[80] Gregory J Chaitin. Register allocation and spilling via graph coloring. *ACM Sigplan Notices*, 17(6):98–101, 1982.

[81] Stefan M Freudenberger, Jacob T Schwartz, and Micha Sharir. Experience with the setl optimizer.

ACM Transactions on Programming Languages and Systems (TOPLAS), 5(1):26–45, 1983.

[82] John L. Hennessy and Thomas Gross. Postpass code optimization of pipeline constraints. *ACM Trans. Program. Lang. Syst.*, 5(3):422–448, July 1983.

[83] John R Allen, Ken Kennedy, Carrie Porterfield, and Joe Warren. Conversion of control dependence to data dependence. In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 177–189, 1983.

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

[85] Jack W Davidson and Christopher W Fraser. Automatic generation of peephole optimizations. In *Proceedings of the 1984 SIGPLAN symposium on Compiler construction*, pages 111–116, 1984.

[86] David Callahan and Ken Kennedy. Analysis of interprocedural side effects in a parallel programming environment. In *International Conference on Supercomputing*, pages 138–171. Springer, 1987.

[87] Mikhail A. Bulyonkov. Polyvariant mixed computation for analyzer programs. *Acta Informatica*, 21(5):473–484, 1984.

[88] Allen Goldberg and Robert Paige. Stream processing. In *Proceedings of the 1984 ACM Symposium on LISP and Functional Programming*, LFP '84, page 53–62, New York, NY, USA, 1984. Association for Computing Machinery.

[89] Joe Warren. A hierarchical basis for reordering transformations. In *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '84, page 272–282, New York, NY, USA, 1984. Association for Computing Machinery.

[90] Frederick Chow and John Hennessy. Register allocation by priority-based coloring. *SIGPLAN Not.*, 19(6):222–232, June 1984.

[91] John R Allen and Ken Kennedy. Automatic loop interchange. In *Proceedings of the 1984 SIGPLAN symposium on Compiler construction*, pages 233–246, 1984.

[92] Keith D Cooper and Ken Kennedy. Efficient computation of flow insensitive interprocedural summary information. In *Proceedings of the 1984 SIGPLAN symposium on Compiler construction*, pages 247–258, 1984.

[93] Jeanne Ferrante and Mary Mace. On linearizing parallel code. In *Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '85, page 179–190, New York, NY, USA, 1985. Association for Computing Machinery.

[94] Paul Hudak and Benjamin Goldberg. Distributed execution of functional programs using serial combinators. *IEEE Transactions on computers*, 100(10):881–891, 1985.

[95] Chris Clack and Simon L Peyton Jones. Strictness analysis-a practical approach. In *Conference on Functional Programming Languages and Computer Architecture*, pages 35–49. Springer, 1985.

[96] D Harel. A linear algorithm for finding dominators in flow graphs and related problems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 185–194, New York, NY, USA, 1985. Association for Computing Machinery.

[97] Philip B Gibbons and Steven S Muchnick. Efficient instruction scheduling for a pipelined architecture. In *Proceedings of the 1986 SIGPLAN symposium on Compiler construction*, pages 11–16, 1986.

[98] John H. Reif and Harry R. Lewis. Efficient symbolic analysis of programs. *Journal of Computer and System Sciences*, 32(3):280–314, 1986.

[99] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[100] Michael Wolfe. Loops skewing: The wavefront method revisited. *International Journal of Parallel Programming*, 15(4):279–293, 1986.

[101] P YT Hsu and Edward S Davidson. Highly concurrent scalar processing. *ACM SIGARCH Computer Architecture News*, 14(2):386–395, 1986.

[102] Robert Bernstein. Multiplication by integer constants. *Software: practice and experience*, 16(7):641–652, 1986.

[103] David W. Wall. Global register allocation at link time. In *Proceedings of the 1986 SIGPLAN Symposium on Compiler Construction*, SIGPLAN '86, page 264–275, New York, NY, USA, 1986. Association for Computing Machinery.

[104] David Callahan, Keith D Cooper, Ken Kennedy, and Linda Torczon. Interprocedural constant propagation. *ACM SIGPLAN Notices*, 21(7):152–161, 1986.

[105] Keith D. Cooper, Ken Kennedy, and Linda Torczon. Interprocedural optimization: eliminating unnecessary recompilation. In *Proceedings of the 1986 SIGPLAN Symposium on Compiler Construction*, SIGPLAN '86, page 58–67, New York, NY, USA, 1986. Association for Computing Machinery.

[106] Michael Burke and Ron Cytron. Interprocedural dependence analysis and parallelization. *ACM Sigplan Notices*, 21(7):162–175, 1986.

[107] Mark S. Johnson and Terrence C. Miller. Effectiveness of a machine-level, global optimizer. In *Proceedings of the 1986 SIGPLAN Symposium on Compiler Construction*, SIGPLAN '86, page 99–108, New York, NY, USA, 1986. Association for Computing Machinery.

[108] Rémi Triolet, Francois Irigoin, and Paul Feautrier. Direct parallelization of call statements. *SIGPLAN Not.*, 21(7):176–185, July 1986.

[109] Ron Cytron, Andy Lowry, and F Kenneth Zadeck. Code motion of control structures in high-level languages. In *Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 70–85, 1986.

[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, 1987.

[111] Samuel P. Midkiff and David A. Padua. Compiler algorithms for synchronization. *IEEE Transactions on Computers*, C-36(12):1485–1495, 1987.

[112] r. Allen, D. Callahan, and K. Kennedy. Automatic decomposition of scientific programs for parallel execution. In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '87, page 63–76, New York, NY, USA, 1987. Association for Computing Machinery.

[113] Constantine D. Polychronopoulos and David J. Kuck. Guided self-scheduling: A practical scheduling scheme for parallel supercomputers. *IEEE Transactions on Computers*, C-36(12):1425–1439, 1987.

[114] Constantine D Polychronopoulos. Loop coalescing: A compiler transformation for parallel machines. Technical report, Illinois Univ., Urbana (USA), 1987.

[115] Dennis Gannon, William Jalby, and Kyle Gallivan. Strategies for cache and local memory management by global program transformation. In *International Conference on Supercomputing*, pages 229–254. Springer, 1987.

[116] 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 (TOPLAS)*, 9(3):319–349, 1987.

[117] Randy Allen and Ken Kennedy. Automatic translation of fortran programs to vector form. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 9(4):491–542, 1987.

[118] S. Jain and C. Thompson. An efficient approach to data flow analysis in a multiple pass global optimizer. *SIGPLAN Not.*, 23(7):154–163, June 1988.

[119] 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 Trans. Program. Lang. Syst.*,

10(4):635–640, October 1988.

[120] Keith D Cooper and Ken Kennedy. Efficient computation of flow-insensitive interprocedural summary information—a correction. *ACM SIGPLAN Notices*, 23(4):35–42, 1988.

[121] D. R. Wallace. Dependence of multi-dimensional array references. In *Proceedings of the 2nd International Conference on Supercomputing*, ICS '88, page 418–428, New York, NY, USA, 1988. Association for Computing Machinery.

[122] T. Brandes. The importance of direct dependences for automatic parallelization. In *Proceedings of the 2nd International Conference on Supercomputing*, ICS '88, page 407–417, New York, NY, USA, 1988. Association for Computing Machinery.

[123] A. Lichnewsky and F. Thomasset. Introducing symbolic problem solving techniques in the dependence testing phases of a vectorizer. In *Proceedings of the 2nd International Conference on Supercomputing*, ICS '88, page 396–406, New York, NY, USA, 1988. Association for Computing Machinery.

[124] J. Ferrante, M. Mace, and B. Simons. Generating sequential code from parallel code. In *Proceedings of the 2nd International Conference on Supercomputing*, ICS '88, page 582–592, New York, NY, USA, 1988. Association for Computing Machinery.

[125] Constantine D. Polychronopoulos. Advanced loop optimizations for parallel computers. In E. N. Houstis, T. S. Papatheodorou, and C. D. Polychronopoulos, editors, *Supercomputing*, pages 255–277, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[126] Utpal Banerjee. An introduction to a formal theory of dependence analysis. *The Journal of Supercomputing*, 2:133–149, 1988.

[127] David Callahan and Ken Kennedy. Analysis of interprocedural side effects in a parallel programming environment. In E. N. Houstis, T. S. Papatheodorou, and C. D. Polychronopoulos, editors, *Supercomputing*, pages 138–171, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[128] Paul Feautrier. Array expansion. In *ACM International Conference on Supercomputing 25th Anniversary Volume*, pages 99–111, 1988.

[129] Alexandru Nicolau. Loop quantization: A generalized loop unwinding technique. *Journal of Parallel and Distributed Computing*, 5(5):568–586, 1988.

[130] François Irigoin and Rémi Triolet. Supernode partitioning. In *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 319–329, 1988.

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

[132] David Callahan and Ken Kennedy. Compiling programs for distributed-memory multiprocessors. *The Journal of Supercomputing*, 2:151–169, 1988.

[133] Alexander Aiken and Alexandru Nicolau. Optimal loop parallelization. *ACM SIGPLAN Notices*, 23(7):308–317, 1988.

[134] M. Lam. Software pipelining: an effective scheduling technique for vliw machines. *SIGPLAN Not.*, 23(7):318–328, June 1988.

[135] David Callahan. The program summary graph and flow-sensitive interprocedual data flow analysis. In *Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation*, pages 47–56, 1988.

[136] Frances Allen, Michael Burke, Ron Cytron, Jeanne Ferrante, and Wilson Hsieh. A framework for determining useful parallelism. In *Proceedings of the 2nd international conference on Supercomputing*, pages 207–215, 1988.

[137] Frances Allen, Michael Burke, Philippe Charles, Ron Cytron, and Jeanne Ferrante. An overview of the ptran analysis system for multiprocessing. *Journal of parallel and distributed computing*, 5(5):617–

640, 1988.

[138] Zhiyuan Li and Pen-Chung Yew. Efficient interprocedural analysis for program parallelization and restructuring. *SIGPLAN Not.*, 23(9):85–99, January 1988.

[139] M. Girkar and C. Polychronoupoulos. Compiling issues for supercomputers. In *Supercomputing '88:Proceedings of the 1988 ACM/IEEE Conference on Supercomputing, Vol. I*, pages 164–173, 1988.

[140] Fred C Chow. Minimizing register usage penalty at procedure calls. In *Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation*, pages 85–94, 1988.

[141] David Callahan, John Cocke, and Ken Kennedy. Estimating interlock and improving balance for pipelined architectures. *Journal of Parallel and Distributed Computing*, 5(4):334–358, 1988.

[142] B. K. Rosen, M. N. Wegman, and F. K. Zadeck. Global value numbers and redundant computations. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '88, page 12–27, New York, NY, USA, 1988. Association for Computing Machinery.

[143] Alexander Aiken and Alexandru Nicolau. Perfect pipelining: A new loop parallelization technique. In *European Symposium on Programming*, pages 221–235. Springer, 1988.

[144] James R Goodman and W-C Hsu. Code scheduling and register allocation in large basic blocks. In *ACM International Conference on Supercomputing 25th Anniversary Volume*, pages 88–98, 1988.

[145] Keith D Cooper and Ken Kennedy. Interprocedural side-effect analysis in linear time. *ACM SIGPLAN Notices*, 23(7):57–66, 1988.

[146] C-H Chi and Hank Dietz. Unified management of registers and cache using liveness and cache bypass. In *Proceedings of the ACM SIGPLAN 1989 conference on Programming language design and implementation*, pages 344–353, 1989.

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

[148] S. McFarling. Program optimization for instruction caches. *SIGARCH Comput. Archit. News*, 17(2):183–191, April 1989.

[149] S. Horwitz, P. Pfeiffer, and T. Reps. Dependence analysis for pointer variables. *SIGPLAN Not.*, 24(7):28–40, June 1989.

[150] W. Baxter and H. R. Bauer. The program dependence graph and vectorization. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, page 1–11, New York, NY, USA, 1989. Association for Computing Machinery.

[151] W-m W Hwu and Pohua P Chang. Achieving high instruction cache performance with an optimizing compiler. In *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pages 242–251, 1989.

[152] Susan J Eggers and Randy H Katz. Evaluating the performance of four snooping cache coherency protocols. In *Proceedings of the 16th annual international symposium on Computer architecture*, pages 2–15, 1989.

[153] Guy E Blelloch. Scans as primitive parallel operations. *IEEE Transactions on computers*, 38(11):1526–1538, 1989.

[154] Alfred V. Aho, Mahadevan Ganapathi, and Steven W. K. Tjiang. Code generation using tree matching and dynamic programming. *ACM Trans. Program. Lang. Syst.*, 11(4):491–516, October 1989.

[155] R. Gupta, M. L. Soffa, and T. Steele. Register allocation via clique separators. In *Proceedings of the ACM SIGPLAN 1989 Conference on Programming Language Design and Implementation*, PLDI '89, page 264–274, New York, NY, USA, 1989. Association for Computing Machinery.

[156] K. D. Cooper and K. Kennedy. Fast interprocedual alias analysis. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, page 49–59, New

York, NY, USA, 1989. Association for Computing Machinery.

[157] P. Briggs, K. D. Cooper, K. Kennedy, and L. Torczon. Coloring heuristics for register allocation. In *Proceedings of the ACM SIGPLAN 1989 Conference on Programming Language Design and Implementation*, PLDI '89, page 275–284, New York, NY, USA, 1989. Association for Computing Machinery.

[158] M. Wolfe. More iteration space tiling. In *Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, Supercomputing '89, page 655–664, New York, NY, USA, 1989. Association for Computing Machinery.

[159] Zhiyuan Li, Pen-Chung Yew, and Chuag-Qi Zhu. Data dependence analysis on multi-dimensional array references. In *Proceedings of the 3rd International Conference on Supercomputing*, ICS '89, page 215–224, New York, NY, USA, 1989. Association for Computing Machinery.

[160] D. Bernstein, M. Golumbic, y. Mansour, R. Pinter, D. Goldin, H. Krawczyk, and I. Nahshon. Spill code minimization techniques for optimizing compliers. In *Proceedings of the ACM SIGPLAN 1989 Conference on Programming Language Design and Implementation*, PLDI '89, page 258–263, New York, NY, USA, 1989. Association for Computing Machinery.

[161] Craig Chambers and David Ungar. Customization: Optimizing compiler technology for self, a dynamically-typed object-oriented programming language. *ACM SIGPLAN Notices*, 24(7):146–160, 1989.

[162] Ron Cytron, Jeanne Ferrante, Barry K Rosen, Mark N Wegman, and F Kenneth Zadeck. An efficient method of computing static single assignment form. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 25–35, 1989.

[163] D. Whitfield and M. L. Soffa. An approach to ordering optimizing transformations. In *Proceedings of the Second ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, PPOPP '90, page 137–146, New York, NY, USA, 1990. Association for Computing Machinery.

[164] Vatsa Santhanam and Daryl Odnert. Register allocation across procedure and module boundaries. *SIGPLAN Not.*, 25(6):28–39, June 1990.

[165] Rajiv Gupta and Mary Lou Soffa. Region scheduling: An approach for detecting and redistributing parallelism. *IEEE Trans. Softw. Eng.*, 16(4):421–431, April 1990.

[166] D. Callahan, A. Carle, M.W. Hall, and K. Kennedy. Constructing the procedure call multigraph. *IEEE Transactions on Software Engineering*, 16(4):483–487, 1990.

[167] David Klappholz, Kleanthis Psarris, and Xiangyun Kong. On the perfect accuracy of an approximate subscript analysis test. *ACM SIGARCH Computer Architecture News*, 18(3b):201–212, 1990.

[168] Thomas Gross and Peter Steenkiste. Structured dataflow analysis for arrays and its use in an optimizing compiler. *Software: Practice and Experience*, 20(2):133–155, 1990.

[169] Steven Anderson and Paul Hudak. Compilation of haskell array comprehensions for scientific computing. *ACM SIGPLAN Notices*, 25(6):137–149, 1990.

[170] William D. Clinger. How to read floating point numbers accurately. *SIGPLAN Not.*, 25(6):92–101, June 1990.

[171] Guy L. Steele and Jon L. White. How to print floating-point numbers accurately. *SIGPLAN Not.*, 25(6):112–126, June 1990.

[172] Karl Pettis and Robert C. Hansen. Profile guided code positioning. *SIGPLAN Not.*, 25(6):16–27, June 1990.

[173] Michael Gerndt. Updating distributed variables in local computations. *Concurrency: Practice and Experience*, 2(3):171–193, 1990.

[174] Michael Burke. An interval-based approach to exhaustive and incremental interprocedural data-flow analysis. *ACM Trans. Program. Lang. Syst.*, 12(3):341–395, July 1990.

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

[176] David Callahan, Steve Carr, and Ken Kennedy. Improving register allocation for subscripted variables. *ACM Sigplan Notices*, 25(6):53–65, 1990.

[177] David R. Chase, Mark Wegman, and F. Kenneth Zadeck. Analysis of pointers and structures. In *Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation*, PLDI '90, page 296–310, New York, NY, USA, 1990. Association for Computing Machinery.

[178] Ken Kennedy and Kathryn S McKinley. Loop distribution with arbitrary control flow. In *SC*, pages 407–416. Citeseer, 1990.

[179] Brian R. Nickerson. Graph coloring register allocation for processors with multi-register operands. *SIGPLAN Not.*, 25(6):40–52, June 1990.

[180] David Callahan and Brian Koblenz. Register allocation via hierarchical graph coloring. *ACM Sigplan Notices*, 26(6):192–203, 1991.

[181] Suneel Jain. Circular scheduling: a new technique to perform software pipelining. In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, PLDI '91, page 219–228, New York, NY, USA, 1991. Association for Computing Machinery.

[182] Mark Smotherman, Sanjay Krishnamurthy, P. S. Aravind, and David Hunnicutt. Efficient dag construction and heuristic calculation for instruction scheduling. In *Proceedings of the 24th Annual International Symposium on Microarchitecture*, MICRO 24, page 93–102, New York, NY, USA, 1991. Association for Computing Machinery.

[183] Dhananjay M Dhamdhere and JR Isaac. A composite algorithm for strength reduction and code movement optimization. *International Journal of Computer & Information Sciences*, 9:243–273, 1980.

[184] 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 Trans. Program. Lang. Syst.*, 13(4):451–490, October 1991.

[185] David Callahan, Ken Kennedy, and Allan Porterfield. Software prefetching. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IV, page 40–52, New York, NY, USA, 1991. Association for Computing Machinery.

[186] C. Koelbel and P. Mehrotra. Compiling global name-space parallel loops for distributed execution. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):440–451, 1991.

[187] Paul Havlak and Ken Kennedy. An implementation of interprocedural bounded regular section analysis. *IEEE Transactions on Parallel & Distributed Systems*, 2(03):350–360, 1991.

[188] David W Wall. Limits of instruction-level parallelism. In *Proceedings of the fourth international conference on Architectural support for programming languages and operating systems*, pages 176–188, 1991.

[189] William Pugh. Uniform techniques for loop optimization. In *Proceedings of the 5th international conference on Supercomputing*, pages 341–352, 1991.

[190] Michael E. Wolf and Monica S. Lam. A data locality optimizing algorithm. *SIGPLAN Not.*, 26(6):30–44, May 1991.

[191] 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.

[192] Dror E. Maydan, John L. Hennessy, and Monica S. Lam. Efficient and exact data dependence analysis. *SIGPLAN Not.*, 26(6):1–14, May 1991.

[193] 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 Trans. Program.*

Lang. Syst., 13(4):451–490, October 1991.

[194] David Bernstein and Michael Rodeh. Global instruction scheduling for superscalar machines. *SIGPLAN Not.*, 26(6):241–255, May 1991.

[195] Dhananjay M. Dhamdhere. Practical adaption of the global optimization algorithm of morel and renvoise. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(2):291–294, 1991.

[196] Monica D. Lam, Edward E. Rothberg, and Michael E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IV, page 63–74, New York, NY, USA, 1991. Association for Computing Machinery.

[197] Michael E Wolf and Monica S Lam. A loop transformation theory and an algorithm to maximize parallelism. *IEEE Transactions on Parallel & Distributed Systems*, 2(04):452–471, 1991.

[198] Paul Feautrier. Dataflow analysis of array and scalar references. *International Journal of Parallel Programming*, 20:23–53, 1991.

[199] Vivek Sarkar and Guang R. Gao. Optimization of array accesses by collective loop transformations. In *Proceedings of the 5th International Conference on Supercomputing*, ICS '91, page 194–205, New York, NY, USA, 1991. Association for Computing Machinery.

[200] Mary W Hall, Ken Kennedy, and Kathryn S McKinley. Interprocedural transformations for parallel code generation. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 424–434, 1991.

[201] Gina Goff, Ken Kennedy, and Chau-Wen Tseng. Practical dependence testing. *ACM SIGPLAN Notices*, 26(6):15–29, 1991.

[202] Scott McFarling. Procedure merging with instruction caches. *ACM SIGPLAN Notices*, 26(6):71–79, 1991.

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

[204] S. Tjiang, M. Wolf, M. Lam, K. Pieper, and J. Hennessy. Integrating scalar optimization and parallelization. In Utpal Banerjee, David Gelernter, Alex Nicolau, and David Padua, editors, *Languages and Compilers for Parallel Computing*, pages 137–151, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[205] Steven W. K. Tjiang and John L. Hennessy. Sharlit—a tool for building optimizers. *SIGPLAN Not.*, 27(7):82–93, July 1992.

[206] Dhananjay M Dhamdhere, Barry K Rosen, and F Kenneth Zadeck. How to analyze large programs efficiently and informatively. In *Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation*, pages 212–223, 1992.

[207] S. A. Mahlke, W. Y. Chen, J. C. Gyllenhaal, and W.-M. W. Hwu. Compiler code transformations for superscalar-based high performance systems. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, Supercomputing '92, page 808–817, Washington, DC, USA, 1992. IEEE Computer Society Press.

[208] Christopher W. Fraser, David R. Hanson, and Todd A. Proebsting. Engineering a simple, efficient code-generator generator. *ACM Lett. Program. Lang. Syst.*, 1(3):213–226, September 1992.

[209] Paul Feautrier. Some efficient solutions to the affine scheduling problem. i. one-dimensional time. *International journal of parallel programming*, 21:313–347, 1992.

[210] Frank Mueller and David B. Whalley. Avoiding unconditional jumps by code replication. In *Proceedings of the ACM SIGPLAN 1992 Conference on Programming Language Design and Implementation*, PLDI '92, page 322–330, New York, NY, USA, 1992. Association for Computing Machinery.

[211] Preston Briggs, Keith D. Cooper, and Linda Torczon. Rematerialization. *SIGPLAN Not.*, 27(7):311–321, July 1992.

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

[213] Ken Kennedy and Kathryn S McKinley. Optimizing for parallelism and data locality. In *Proceedings of the 6th international conference on Supercomputing*, pages 323–334, 1992.

[214] Ken Kennedy and Kathryn S McKinley. Maximizing loop parallelism and improving data locality via loop fusion and distribution. In *International Workshop on Languages and Compilers for Parallel Computing*, pages 301–320. Springer, 1993.

[215] Michael Wolfe. Beyond induction variables. *SIGPLAN Not.*, 27(7):162–174, July 1992.

[216] Mary W. Hall and Ken Kennedy. Efficient call graph analysis. *ACM Lett. Program. Lang. Syst.*, 1(3):227–242, September 1992.

[217] William Landi and Barbara G. Ryder. A safe approximate algorithm for interprocedural aliasing. In *Proceedings of the ACM SIGPLAN 1992 Conference on Programming Language Design and Implementation*, PLDI '92, page 235–248, New York, NY, USA, 1992. Association for Computing Machinery.

[218] Reinhard v. Hanxleden and Ken Kennedy. Relaxing simd control flow constraints using loop transformations. *SIGPLAN Not.*, 27(7):188–199, July 1992.

[219] Keith D. Cooper, Mary W. Hall, and Linda Torczon. Unexpected side effects of inline substitution: a case study. *ACM Lett. Program. Lang. Syst.*, 1(1):22–32, March 1992.

[220] William Pugh and David Wonnacott. Eliminating false data dependences using the omega test. *SIGPLAN Not.*, 27(7):140–151, July 1992.

[221] Jens Knoop, Oliver Rüthing, and Bernhard Steffen. Lazy code motion. *SIGPLAN Not.*, 27(7):224–234, July 1992.

[222] Anne Rogers and Kai Li. Software support for speculative loads. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS V, page 38–50, New York, NY, USA, 1992. Association for Computing Machinery.

[223] William Pugh. A practical algorithm for exact array dependence analysis. *Commun. ACM*, 35(8):102–114, August 1992.

[224] Zhiyuan Li. Array privatization for parallel execution of loops. In *Proceedings of the 6th International Conference on Supercomputing*, ICS '92, page 313–322, New York, NY, USA, 1992. Association for Computing Machinery.

[225] Michael Wolfe, Chau-Wen Tseng, et al. The power test for data dependence. *IEEE Transactions on Parallel and Distributed Systems*, 3(5):591–601, 1992.

[226] Torbjörn Granlund and Richard Kenner. Eliminating branches using a superoptimizer and the gnu c compiler. In *Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation*, pages 341–352, 1992.

[227] Todd C. Mowry, Monica S. Lam, and Anoop Gupta. Design and evaluation of a compiler algorithm for prefetching. *SIGPLAN Not.*, 27(9):62–73, September 1992.

[228] Richard Johnson and Keshav Pingali. Dependence-based program analysis. In *Proceedings of the ACM SIGPLAN 1993 Conference on Programming Language Design and Implementation*, PLDI '93, page 78–89, New York, NY, USA, 1993. Association for Computing Machinery.

[229] Karl-Heinz Drechsler and Manfred P Stadel. A variation of knoop, rüthing, and steffen's lazy code motion. *ACM SIGPLAN Notices*, 28(5):29–38, 1993.

[230] Jens Knoop. Lazy strength reduction. *International Journal of Programming Languages*, 1(1):71–91, 1993.

[231] G. Gao, R. Olsen, V. Sarkar, and R. Thekkath. Collective loop fusion for array contraction. In Utpal Banerjee, David Gelernter, Alex Nicolau, and David Padua, editors, *Languages and Compilers for Parallel Computing*, pages 281–295, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[232] Robert Metzger and Sean Stroud. Interprocedural constant propagation: an empirical study. *ACM Lett. Program. Lang. Syst.*, 2(1–4):213–232, March 1993.

[233] Mohammad R Haghighat and Constantine D Polychronopoulos. Symbolic analysis: A basis for parallelization, optimization, and scheduling of programs. In *International Workshop on Languages and Compilers for Parallel Computing*, pages 567–585. Springer, 1993.

[234] Dror E. Maydan, Saman P. Amarasinghe, and Monica S. Lam. Array-data flow analysis and its use in array privatization. In *Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '93, page 2–15, New York, NY, USA, 1993. Association for Computing Machinery.

[235] Siddhartha Chatterjee, John R Gilbert, Robert Schreiber, and Shang-Hua Teng. Automatic array alignment in data-parallel programs. In *Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 16–28, 1993.

[236] Jennifer M Anderson and Monica S Lam. Global optimizations for parallelism and locality on scalable parallel machines. *ACM Sigplan Notices*, 28(6):112–125, 1993.

[237] B. Ramakrishna Rau and Joseph A. Fisher. *Instruction-Level Parallel Processing: History, Overview, and Perspective*, pages 9–50. Springer US, Boston, MA, 1993.

[238] Amitabh Srivastava. A practical system for intermodule code optimization at link-time. *Journal of programming Languages*, 1(1):1–18, 1993.

[239] James R Larus. Loop-level parallelism in numeric and symbolic programs. *IEEE Transactions on Parallel and Distributed Systems*, 4(7):812–826, 1993.

[240] Susan L. Graham, Steven Lucco, and Oliver Sharp. Orchestrating interactions among parallel computations. *SIGPLAN Not.*, 28(6):100–111, June 1993.

[241] 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, et al. The superblock: an effective technique for vliw and superscalar compilation. *The Journal of Supercomputing*, 7(1-2):229–248, 1993.

[242] Rajiv Gupta. Optimizing array bound checks using flow analysis. *ACM Lett. Program. Lang. Syst.*, 2(1–4):135–150, March 1993.

[243] Shlomit S. Pinter. Register allocation with instruction scheduling. *SIGPLAN Not.*, 28(6):248–257, June 1993.

[244] Keith D Cooper, Mary W Hall, and Ken Kennedy. A methodology for procedure cloning. *Computer Languages*, 19(2):105–117, 1993.

[245] Jong-Deok Choi, Michael Burke, and Paul Carini. Efficient flow-sensitive interprocedural computation of pointer-induced aliases and side effects. In *Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 232–245, 1993.

[246] Peng Tu and David Padua. Automatic array privatization. In *International Workshop on Languages and Compilers for Parallel Computing*, pages 500–521. Springer, 1993.

[247] William Landi, Barbara G Ryder, and Sean Zhang. Interprocedural modification side effect analysis with pointer aliasing. *ACM SIGPLAN Notices*, 28(6):56–67, 1993.

[248] Thomas Ball and James R Larus. Branch prediction for free. *ACM SIGPLAN Notices*, 28(6):300–313, 1993.

[249] Amitabh Srivastava and David W. Wall. Link-time optimization of address calculation on a 64-bit

architecture. In *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, PLDI '94, page 49–60, New York, NY, USA, 1994. Association for Computing Machinery.

[250] Preston Briggs and Keith D Cooper. Effective partial redundancy elimination. *ACM SIGPLAN Notices*, 29(6):159–170, 1994.

[251] Jens Knoop, Oliver Rüthing, and Bernhard Steffen. Partial dead code elimination. *ACM Sigplan Notices*, 29(6):147–158, 1994.

[252] Joseph Hummel, Laurie J. Hendren, and Alexandru Nicolau. A general data dependence test for dynamic, pointer-based data structures. In *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, PLDI '94, page 218–229, New York, NY, USA, 1994. Association for Computing Machinery.

[253] Uma Mahadevan and Sridhar Ramakrishnan. Instruction scheduling over regions: A framework for scheduling across basic blocks. In Peter A. Fritzson, editor, *Compiler Construction*, pages 419–434, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[254] Daniel Weise, Roger F. Crew, Michael Ernst, and Bjarne Steensgaard. Value dependence graphs: representation without taxation. In *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '94, page 297–310, New York, NY, USA, 1994. Association for Computing Machinery.

[255] Steven M. Kurlander and Charles N. Fischer. Zero-cost range splitting. *SIGPLAN Not.*, 29(6):257–265, June 1994.

[256] Brad Calder and Dirk Grunwald. Reducing branch costs via branch alignment. *SIGOPS Oper. Syst. Rev.*, 28(5):242–251, November 1994.

[257] Steve Carr and Ken Kennedy. Improving the ratio of memory operations to floating-point operations in loops. *ACM Trans. Program. Lang. Syst.*, 16(6):1768–1810, November 1994.

[258] Eric Amiel, Olivier Gruber, and Eric Simon. Optimizing multi-method dispatch using compressed dispatch tables. *SIGPLAN Not.*, 29(10):244–258, October 1994.

[259] Cliff Young and Michael D. Smith. Improving the accuracy of static branch prediction using branch correlation. *SIGOPS Oper. Syst. Rev.*, 28(5):232–241, November 1994.

[260] David F. Bacon, Jyh-Herng Chow, Dz-Ching Ju, Kalyan Muthukumar, and Vivek Sarkar. A compiler framework for restructuring data declarations to enhance cache and TLB effectiveness. In John E. Botsford, Ann Gawman, W. Morven Gentleman, Evelyn Kidd, Kelly A. Lyons, Jacob Slonim, and J. Howard Johnson, editors, *Proceedings of the 1994 Conference of the Centre for Advanced Studies on Collaborative Research, October 31 - November 3, 1994 , Toronto, Ontario, Canada*, page 3. IBM, 1994.

[261] J. Torrellas, H.S. Lam, and J.L. Hennessy. False sharing and spatial locality in multiprocessor caches. *IEEE Transactions on Computers*, 43(6):651–663, 1994.

[262] Peter W Markstein, Victoria Markstein, and F Kenneth Zadeck. Reassociation and strength reduction. *Optimization in Compilers*, 1994.

[263] Siddhartha Chatterjee, John R Gilbert, and Robert Schreiber. The alignment-distribution graph. In *Languages and Compilers for Parallel Computing: 6th International Workshop Portland, Oregon, USA, August 12–14, 1993 Proceedings 6*, pages 234–252. Springer, 1994.

[264] Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to graph coloring register allocation. *ACM Trans. Program. Lang. Syst.*, 16(3):428–455, May 1994.

[265] Maryam Emami, Rakesh Ghiya, and Laurie J. Hendren. Context-sensitive interprocedural points-to analysis in the presence of function pointers. In *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, PLDI '94, page 242–256, New York, NY, USA,

1994. Association for Computing Machinery.

[266] Alain Deutsch. Interprocedural may-alias analysis for pointers: beyond k-limiting. In *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, PLDI '94, page 230–241, New York, NY, USA, 1994. Association for Computing Machinery.

[267] Steve Carr and Ken Kennedy. Scalar replacement in the presence of conditional control flow. *Softw. Pract. Exper.*, 24(1):51–77, January 1994.