HOW TO SEARCH EFFICIENTLY*

Cynthia A* Brown and Paul Walton Purdom, Jr.

Computer Science Department
Indiana University
Bloomington, IN M7405

ABSTRACT

The only technique available for solving many important problems is searching. Since searching can be extremely costly, it is important to identify features that improve the efficiency of aearch algorithms. We compute the efficiency of simple backtracking, of an algorithm similar to backtracking except that it notices when the predicate is empty instead of noticing when it is unsatlsfiable (the empty predicate method), of the combination of simple backtracking with the empty predicate method, and of search rearrangement backtracking (the unit clause rule combined with backtracking). The analysis is done over two sets of random problems. We also consider the algorithm based on the pure literal rule that was analyzed by Goldberg and the results he obtained. All these algorithms are simplifications of the complete Putham-David procedure, which has not been analyzed as yet (although most of its components have been analyzed). The performances of the algorithms are compared and features that lead to efficient algorithms are identified.

1. Introduction

Some problems can be aolved by direct calculation in an efficient, straightforward way, but for many important classes of problems the best known method is a controlled search for solutions. Such searches, unfortunately, can consume extremely (exponentially) large amounts of time. Efforts to study and improve search methods are therefore of considerable practical importance.

By carefully analyzing a particular set of problems it may be possible to find problem-specific information that can be used to control the search. This can be an excellent approach; in some cases it has led to algorithms that avoid searching altogether. Often, however, after all problem-specific information has been used, excessive search time is still required. Another approach is to study general search algorithms and identify features that lead to an efficient search. The two approaches are complementary; the best algorithm for a particular problem is often obtained by combining problem-specific techniques with the best general search methods.

Here we report the initial results of a systematic study of the average time performance of search methods. All methods have about the same worst case time (exponentially large). Also, techniques that lead to an improvement in average performance often result in a minor degradation of worst-case behavior, so a study of worst-case behavior can be misleading. The average time performance of these algorithms can be much better than the worst-case performance. Some of the methods we study lead to an exponential improvement in average search time.

The search methods we analyzed include simple backtracking, search rearrangement backtracking [1,15], an algorithm similar to backtracking except that it notices when the predicate is true instead of noticing when it is unsatlsfiable (the empty predicate method), and the empty predicate method combined with simple backtracking. Our analysis shows that the empty predicate rule does not contribute much to the performance of search algorithms, and that search rearrangement backtracking can be much more efficient than ordinary backtracking when the typical problem contains a large number of clauses with few literals per clause.

Goldberg [8] analyzed a version of the Putnam-

* This research was supported in part by the National Science Foundation under Grant no. MCS 7906110.

Davis procedure that essentially relies on the pure literal rule. The analysis shows that the pure literal rule can save a huge amount of time on problems that have a large number of literals per clause, but it also suggests that the rule is unimportant when the typical clause does not have many literals.

We hope to analyze the full Putnam-Davis procedure[4]. The points that remain to be analyzed are the effect of stopping the search when the first solution is found and the effect of the pure literal rule when the typical clause has only a few literals. The effect of combining various techniques also needs to be analyzed.

2. Search Procedures

Searching is used to solve problems that can be expressed in the form

$$P = \bigwedge_{1\le i\le m} R_i(v_{a_{i1}},\dots,v_{a_{ij_i}}) \qquad (1)$$

where each R_i, $1\le i\le m$, is a relation (a function whose value is true or false) over a small number of variables, and the variables $v_{a_{ik}}$ are taken from a set $\{v_i \mid 1\le i\le n\}$ of variables, each of which is restricted to a finite set of values. A solution of the problem is an assignment of values to the variables that makes all of the relations true.

Any problem in the class NP can be expressed as a predicate in the form of Eq.O). Many examples of such problems are given in [6]. We illustrate the encoding of problems in this form with the game of generalized instant insanity [16]. The game is played with n cubes. Each face of each cube is painted with some color. The object of the game is to form a stack of n cubes with each cube oriented so that each face of the stack consists of cube faces that have distinct colors. Each cube has twenty-four possible orientations. Since the order of the cubes in the stack is irrelevant, the problem is equivalent to the predicate

$$\bigwedge_{\substack{1\le i,j\le n\\ i\ne j}} R_{ij}(o_i,o_j) \qquad (2)$$

where o_k is the orientation of cube k and $R_{ij}(o_i, O_j)$ is true if and only if, when cube i has orientation o_i and cube j has orientation o_j, the pair of cubes forms a legal stack of height two (all faces of the stack are made up of distinct colors).

The most obvious method of searching for solutions to a predicate in this form is to try all combinations of values of the variables. This sort of exhaustive searoh is prohibitively slow for large problems. Fortunately, the special form of Eq, 1 permits three types of improvements. First, since each relation is defined over a small subset of the variables, a relation may become false as soon as all of its variables have been assigned values. In this case no extension of the current partial assignment of values to variables can be a solution. Such extensions need not be investigated. This is the idea behind backtracking,

A second improvement consists of looking for variables that should be assigned values early in the search. It is particularly helpful to find a variable all of whose values make a clause false (under the current partial assignment), or which has only one value which does not make a clause false. This is the basic idea of search rearrangement backtracking [1,15], and of the unit clause rule in the Putnam-Davis procedure [4],

A third approach Involves looking for the values of a variable that are most likely to lead to a solution. In some cases there is a value that makes all relations which depend on the variable true. In that case only that one value of the variable requires consideration. This is the basis of the pure literal rule in the Putnam-Davis procedure.

These ideas have been used extensively to improve the average running time of search algorithms. We are studying algorithms that use various combinations of the ideas. To specify these algorithms we first give a common general procedure and then present the details that distinguish the various methods. The procedure uses a stack to keep track of which variables have been set.

Generalized Search Procedure

1. [Initialize,] Set each variable to undefined. Set Stack to empty,

2. [Select variable,] Select as the current variable a variable that needs to be tested. If there is no such variable, go to Step 5* Push the current variable onto Stack and mark all of its values as untested.

3. [Select value.] For the current variable select an untested value that requires testing. If there is no such value, go to Step 6. Otherwise set the variable to that value and mark the value as tested.

4. [Test.] If some clause of the predicate is false under the present partial assignment of values to variables, go to Step 3 If all solutions are desired, go to Step 2.

5. [Solution.] The current assignment of values to variables Is a solution (any remaining unasslgned variables may take on any of their values). If only one solution is desired, stop. If all solutions are desired, go to step 3.

6. [Back up.] Set the current variable to undefined. Pop Stack. The new current variable is at the top of Stack. If Stack is empty, stop. Otherwise go to Step 3.

Many search algorithms simplify the predicate as they search. For example, in the Putnam-Davis procedure clauses that become true are dropped. The dropped clauses are restored when the algorithm backs up.

The algorithms that we have analyzed search for all solutions; they never stop at Step 5. In the future, unless otherwise stated, we will assume that Step 5 always goes to Step 3. For these algorithms the order in which values are tested at Step 3 is immaterial, since all values must eventually be tested.

Simple backtracking Is obtained from the generalized search procedure by selecting the variables in a fixed order at Step 2 and the values in a fixed order at Step 3* Every variable and every value that is considered at Step 3 requires testing. In search rearrangement backtracking [1] a simple test is used to select a variable at Step 2. Each value of each variable is tested (using the same test used in Step 4), and the variable for which the fewest values pass the test is selected. More sophisticated search rearrangement algorithms [15] test combinations of values.

In the Putnam-Davis procedure variables are selected by examining the predicate directly rather than by testing the values of variables. This method of selection is more powerful, but it is also more difficult to program and requires more knowledge of the internal structure of the clauses.

The original Putnam-Davis procedure was designed for predicates in conjunctive normal form. To describe a more general procedure we first give some (nonstandard) definitions. A relation R is a <u>unit clause</u> if, under the current partial assignment of values to variables, there is an unset variable v such that R is false for every value but one of v. We call v the <u>associated variable</u> of the unit clause. A variable v is <u>relevant</u> if there is a relation R whose value under the current partial assignment depends on v. A variable v is associated with a BUS literal If under the current partial assignment there is some value x of v such that when v is assigned value x every R that depends on v is true. The value x is called a safe IftlUft*

The generalized Putnam-Davis procedure is our general search procedure with the following modifications. In Step 2, if possible, select a variable associated with a unit clause. Otherwise, if possible, select a variable associated with a pure literal. Only one safe value is tested. If there are no variables associated with pure literals or unit clauses, then select any relevant variable. If there are no relevant variables, go to Step 5. The Putnam-Davis procedure stops when the first solution is found.

We also considered the empty predicate method, an algorithm that may be viewed as an extremely simplified version of the Putnam-Davis procedure. In this algorithm variables are selected in fixed order at Step 2, as long as some variable is relevant (the selected variable is not necessarily relevant). All values are tested at Step 3, and no tests are done at Step 1 (i.e. the test gives the result true) unless all relevant variables have been assigned values. The procedure searches for all solutions. The procedure is like backtracking except that it backs up when all the R_A are true instead of when one of them is false. It works by noticing when the predicate can be simplified to the point where it is empty.

3* <u>Random</u> Prpfrlfifflff

To do an average time analysis it is necessary to select a set of representative problems and a probability distribution over the set. For backtracking it is not obvious what a "typical" problem is. In this paper we consider two types of random problem sets. Both are Instances of conjunctive normal form predicates. For each type of problem set we give a method of forming a random clause; a random predicate is then the conjunction of t random clauses selected independently (thus, a random predicate may happen to contain two copies of the same clause).

In the first method of constructing random clauses each clause has s literals for some fixed s. The s literals are independently selected from the $2v$ possible literals. This method was used in our earlier analyses of backtracking algorithms [2,13].

In the second method each literal has probablity p of being in the clause for some fixed p. This method was used by Goldberg [8]. The two models are roughly equivalent when the parameters are set so that p *z &/2v.*

We follow the original papers in computing "running time" in the two models. In our model we assume that the "running time" is equal to the number of binary nodes in the search tree. The actual running time increases more rapidly (by a factor of approximately v [3]) but this error is unimportant compared to the exponential differences in the average running times of the various search algorithms.

In Goldberg's model we assume that the time to process a node is equal to avt, where a is a constant, v is the number of unset variables, and t is the number of terms that are still being considered. Both unary and binary nodes are counted.

4. Results

Table 1 summarizes the exact results for the average "running time" of various search algorithms. Most of the exact results are not in closed form, so it is difficult to understand their significance. An asymptotic analysis makes these results easier to interpret. To obtain an interesting asymptotic analysis, careful consideration must be given to how s, p, and t should vary as v increases. We believe that

Table 1. Exact formulae for tha number of solutions a&d for tha "running tine" of various starching algorithms for two sodala of random problems.

Case	Our Model	Goldberg's Model
Solutions	$2^v(1-F(v))^t$	
	$F(i) = \left(\frac{i}{2v}\right)^s$	$F(i) = (1-p)^{2v-i}$
Simple Backtracking (Level 0)	$R(0) + \sum_{1\le i\le v} 2^i R(i)\,(1-F(i-1))^t$	
	$R(i) = 1$	$R(i) = a(v-i)t$
Empty Predicate Method	$1 + \sum_{1\le i\le v} 2^i \left[1-\left(1-\left(\frac{i-1}{2v}\right)^s\right)^t\right]$	$at\frac{[2(1-p)]^{v+1} - 2(1-p) + (2p-1)v}{(2p-1)^2}$
Simple Backtracking with Empty Predicate Rule	$1 + \sum_{1\le i\le v} 2^i \left[(1-F(i-1))^t - \left(1-\left(\frac{i-1}{2v}\right)^s\right)^t\right]$	$at\left[v + \sum_{1\le i\le v} \frac{2^i (v-i) H_i}{(1-(1-p)^{2v})^t}\right]$ where $H_i = \prod_{1\le j<i} (1-(1-p)^{2(v-j)})(1-p)^i$
Search Rearrangement Backtracking (Level 1)	See [13].	Modify the results in [13].

keeping s fixed (or letting p= a/v for fixed a) and letting t = by for fixed b gives results similar to those for many interesting realistic problems. This keeps the individual terms small while letting the number of terms increase with v. We also consider t s v for fixed a>1. (For example, generalized instant insanity can be coded with s s 10 (five logical variables for each o^ in the original problem) and t s $(v/5)^2$.) The first choice for t produces problems where the number of constraints increases proportionately to the number of variables. The second choice produces problems where the number of constraints increases more rapidly.

Table 2 gives the approximate value of the logarithm of the average "running time" for each algorithm. The two models generate problems with the same number of solutions when a = (ln 2) s. Usually the form of the answers is the same for the two models, but Goldberg's model generates problems that are much easier to solve by backtracking when t s v (for $1<a<s$). The results show that the empty predicate method is not helpful when p -> 0. A little thought suggests that this method is less helpful than stopping the search when the first solution is found, because both approaches need a solution before they can save any time.

Comparing the results for simple backtracking with those for search rearrangement on our model with t s v° shows that search rearrangement saves about as much time as reducing the size of the clauses by one literal. This exponential improvement indicates that search rearrangement can be much more effective than simple backtracking. Further analysis is needed to determine how search rearrangement behaves for t = bv.

Table 2. The approximate value of the logarithm (base e) of the number of solutions and "running time". The leading term in an asymptotic expansion is given.

Case	Our Model	Goldberg's Model
Simple Backtracking	$(s-1)\left(\frac{2\ln 2}{a}\right)^{\frac{a}{s-1}} v^{\frac{s-a}{s-1}}$	$(a+k)\ln v$ where $1 \le k \le 2$
	$\left(2\left(\frac{2\ln 2}{b(s-1)}\right)^{\frac{1}{s-1}}\ln 2 + b\ln\left(1-\left(\frac{2\ln 2}{b(s-1)}\right)^{\frac{s}{s-1}}\right)\right) v$	$\left[2\ln 2 - \frac{\ln 2}{a}\ln\left(1+\frac{ab}{\ln 2}\right) - b\ln\left(1+\frac{\ln 2}{ab}\right)\right] v$
Empty Predicate Method	$v\ln 2$	$v\ln 2$
	$v\ln 2$	$v\ln 2$
Simple Backtracking with Empty Predicate Rule	$(s-1)\left(\frac{2\ln 2}{a}\right)^{\frac{a}{s-1}} v^{\frac{s-a}{s-1}}$	$(a+k)\ln v$ where $1 \le k \le 2$
	$\left(2\left(\frac{2\ln 2}{b(s-1)}\right)^{\frac{1}{s-1}}\ln 2 + b\ln\left(1-\left(\frac{2\ln 2}{b(s-1)}\right)^{\frac{s}{s-1}}\right)\right) v$	$\left[2\ln 2 - \frac{\ln 2}{a}\ln\left(1+\frac{ab}{\ln 2}\right) - b\ln\left(1+\frac{\ln 2}{ab}\right)\right] v$
Search Rearrangement Backtracking (Level 1)	$O\left(v^{\frac{s-a-1}{s-2}}\right)$	

The following table shows the results of setting $s = 3, a = 3 \ln 2 : 2.08$, and $b = (\ln 2)(\ln(1-2^{-8}))^{-1} z$ 5.19 in the formulas from Table 2. These values for the parameters lead to an interesting set of difficult problems where the typical problem has about one solution regardless of the size of the problem, as is often the case for realistic difficult problems. The values in the table demonstrate clearly the exponential improvement that can result from using simple backtracking.

Case	Our Model	Goldberg's Model
Solutions	≈ 0	≈ 0
Simple Backtracking	0.247 v	0.456 v
Empty Predicate Method	0.693 v	0.693 v
Simple Backtracking with Empty Predicate Rule	0.247 v	0.456 v

5. Goldberg*a results

In [8] Goldberg reported an average time analysis of an algorithm based on using the pure literal rule to simplify the predicate. When no pure literal is available it chooses a variable and creates two simplified predicates, one for each value of the variable. It uses the same stopping rule as the empty predicate method. The analysis is over the Goldberg model as described above, but with p constant. We are preparing a joint paper with Goldberg in which some minor *errors* in the derivation presented in [8] are corrected. (Reference [14] is becoming a draft of this paper.) These analyses show that Goldberg's procedure takes polynomial average time on his model. When p is fixed, the size of the clauses increases with the number of variables, and the number of solutions to a typical predicate also increases. The analysis does not apply to models with variable p, but it suggests that, for the case in which p approaches zero as v becomes large, the performance of Goldberg's algorithm is not dramatically better than that of the empty predicate method.

6. conclusions

The Putnam-Davis procedure is the source of many good ideas for improving the efficiency of search algorithms. It may be viewed as a combination of 1) backtracking, 2) unit clause selection (one level aearoh rearrangement), 3) empty predicate detection, 4) pure literal selection, and 5) stopping at the first solution. The methods of [2,131 and of this paper are adequate to analyze an algorithm with the first three features. To analyze the first four is more difficult. Goldberg [8] analyzed the effect of the pure literal rule in an algorithm where it does not affect the order of selection of the variables. It is quite likely that his methods can be combined with those of [2,13,14] to analyze an algorithm that uses all of the first four features. No work has yet been done on the effect of stopping at the first solution.

The analyses of random problems show that backtracking is effective on problems that contain a large number of clauses with few literals per clause. When backtracking is effective, backtracking combined with the unit clause rule is even more effective, but when backtracking is not effective adding the unit clause rule does not help much. The pure literal rule works well when each clause contains a large number of literals. Since such predicates usually have a large number of solutions, stopping at the first solution is also effective in this case [5]. None of the analyzed methods is very effective for problems with a moderate number of literals per term and with the number of terms equal to several times the number of variables.

The most straightforward direction for future work is completion of the analysis of search rearrangement backtracking. The blank entries in Table 3 for level one backtracking can be filled in by completing calculations similar to those which led to the first entry. The multi-level backtracking algorithms appear to be even more efficient for large problems [3], but analyzing their performance is difficult.

Backtracking algorithms are easy to use: once the general search algorithm has been coded the user need only provide the routine to test partial solutions. Adding features of the Putnam-Davis procedure that manipulate the predicate requires more programming effort, but the resulting algorithm may also be much more powerful. Analyses are needed to determine whether this is the case.

There may be algorithms that are both simpler to analyze and more powerful than the Putnam-Davis procedure. One weakness of the Putnam-Davis procedure is that it does not have any guidance concerning which variable to select in eases where the pure literal rule ami the unit clause rule oo not apply. A good technique is to select a variable from the shortest remaing clauae. A recent analysis by Monien et al [12] shotted that a problem with three literals per term can always be salved in time 1.62^v using an improvement on the Putnam-Davis algorithm that selects variables from the shortest clause. Some improvement is also obtained for problems with more than three literals per term. Other interesting techniques for modifying backtracking have been proposed [7,9,10,11].

Another area where progress is possible is in the use of rules to manipulate the predicate. Subsumption can be combined with the Putnam-Davis procedure. Substitution of equal quantities can also be used.

Many of the techniques used in this paper were developed in earlier work [2,8,13]. Each of the original papers analyses one algorithm on one model. Here we apply the techniques to a variety of algorithms and models to determine how important various features are to efficient searching. A large number of ideas have been suggested for improving the efficiency of searching. If they were all combined the result would be a large, complex program, containing many parts that made little or no contribution to its efficiency. Analysis of average running time is a powerful technique for determining the value of proposed improvements.

REFERENCES

1. Janes R. Bitner end Edward M. Reingold, Backtrack Programming Techniques, Comn, ACM 18 (1975) pp. 651-655.

2. Cynthia A. Brown and Paul Walton Purdom, Jr., An Average Time Analysis of Backtracking, 5IAM JL £flAJLu to appear.

3. Cynthia A. Brown and Paul Walton Purdom, Jr., An Empirical Comparison of Backtracking Algorithms, Indiana University Computer Science Department Technical Report No. 100 (1981).

4. Hartin Davis, George Logemann, and Donald Uveland, A Machine Program for Theorem Proving, &MU.ACM 5 (1962) pp. 394-397.

5. John Franco, Case Western Reserve University, private communication.

6. Michael R. Garey and David &• Johnson, Computers Art Intractability, W. H. Freeman (1979).

7. John Gaachnlg, Performance Measure and Analysis fil certain search Algorithms, Ph.D. Thesis, Carnegie-Mellon University (1979).

8. Allen Goldberg, Average Case Complexity of the Satisfiability Problem, Proceadlnfs of *THE* jEfiuctb Hftdahop $Q Automated Deduction (1979), pp. 1-6.

9. Robert M. Haralick, Larry S. Davis, Azriel Rosenfeld, and David L. Milgram, Reduction Operations for Constraint Satisfaction, Inforfliatiop Sciences 14 (1978), pp. 199-219.

10. Robert M. Haralick and Linda G. Shapiro, The Consistent Labelling Problem: Part I, IEEE Transactions an faktsxn Analysis and fl&cMne Intelligence 1 (1979), pp. 199-219, and The Consistent Labelling Problem: Part II, IEEE Transactions *sm* f&fcfcflcn Analysis ami Hadiing Intelligence 2 (1980), pp. 193-203.

11. Alan K. Mackworth, Consistency in Networks of Relations, Artificial Intelligence 8 (1977), pp. 99-118.

12. Burkhard Monien, Ewald Speckenmeyer, and G. H. Paderborn, 3-Satisfiability is Testable in $0(1.62^r)$ Steps, Bericht Nr. 3/1979, Reihe Theoretische Informstik (1979).

13* Paul Walton Purdom, Jr. and Cynthia A. Brown, An Analysis of Backtracking With Search Rearrangement, Indiana University Computer Science Department Technical Report No. 89 (1980).

14. Paul Walton Purdom, Jr. and Cynthia A. Brown, Average Time Analyses of Simplified Putnam-Davis Procedures, Indiana University Computer Science Department Technical Report No. 101 (1981).

15. Paul Walton Purdom, Jr., Cynthia A. Brown and Edward L. Robertson, Backtracking with Multiple-Level Search Rearrangement, ACTA Informatics 15 (1981), pp. 99-113.

16. Edward Robertson and Ian Munro, NP Completeness, Puzzles and Games, utilitaa Math. 13 (1978), pp. 99-116.

AUTHOR INDEX

Abe, Norihiro. 77
Agar, Michael. 190
Aggarwal, J. K. 686
Agusa, Kiyoshi. 949
Aikins, Janice S. 886
Allen, Elizabeth. 983
Allen, Janes F. 221
Anderson, John R. 97, 163
Arens, Yigal. 52
Asada, Haruhiko. 773
Attardi, Giuseppe • •. 304
Austin, Howard. 846

Baker, Harlyn. 631
Ballard, Dana H. 607, 1068
Bane, Bob. 933
Banerji, Ranan. 127
Barnett, Jeffrey A. 866
Barstov, David R. 927
Bartels, Ulrich. 1037
Barth, Paul. 975
Bechtel, Robert J. 1053
Bennett, James S.. 843
Berliner, Hans J. 561
Berry, Michael. 1054
Bienkowski, M.A. 362
Binford, Thomas 0. . . . 613, 631, 752
Birnbaum, Lawrence. 58
Bischoff, Miriam B. 876
Black, John B. 184
Blasius, K. 511
Bobrow, Daniel G. 913
Boguraev, B.K. 443
Boissonnat, J.D. 658, 796
Bolles, Robert C. 637
Bond, Alan. 159
Borgida, Alexander. 254
Borning, Alan. 466
Bouchard, Susan A. 1065
Brachman, Ronald J. 452
Bradshaw, Gary L. 121
Brooks, Rodney A. 619
Brown, Cynthia A. 588
Brown, Richard H. 998
Buchstaller, Walter. 850
Bundy, Alan. 466, 551

*** indicates that the paper was not received in time for publication

Volume Two begins on page 395

Canmarata, Stephanie. 171
Campbell, A. Bruce. 876
Carbonell, Jaime G. 147, 432
Carlbom, Ingrid. 846
Cavthorn, R.C. 109
Chandrasekaran, B. 1055
Charniak, Eugene. 1079
Clancey, William J. 829
Cohen, Philip R. 31
Coleman, E. North Jr. 652
Collins, Carter. 704
Colmerauer, A. 947, 1056
Coulon, Daniel. 64
Criacuolo, Giovanni. 270
Croucher, Monica. 197
Cullingford, Richard E. 362

Dahl, Hartvig. 394
Davis, Martin. 330
Davis, Randall. 846
de Bruin, Jos. . . . .• • 519
de Champeaux, Dennis 519
DeJong, Gerald. 67
Deering, Michael F. 704, 930
Dehn, Natalie. 16
Descotte, Yannick. 766
Digricoli, Vincent J. 539
Dixon, John K. 1065
Dreschler, L. 692
Dyer, Michael G. 37, 234, 1057

Eisenstadt, Marc. 964
Eisinger, Norbert. 480, 511
Erman, Lee. 409

Fahlman, Scott E. 257
Faletti, Joseph. 930
Faugeras, 0.D. 656
Fickas, Stephen. 409
Firschein, 0. 740
Fischler, Martin A. . . 319, 637, 740
Flowers, Margot •. 58
Forbus, Kenneth D. 326
Fox, Mark S. 313, 1058
Frawley, Bud. 846
Freitas, Robert A. Jr. 803
Friedland, Peter E. 856
Friedman, Leonard. 467
Fu, King-Sun. 837
Funt, Brian V. 216
Furugori, Teiji. . . . . .• • • • 426
Furukawa, Koichi. 1010

Galen, Robert 8. 853
Garvey, Thomas D. 319
Gennery, Donald B. . . . . .• . • 667
Georgeff, H. 563
Germain. F. 796
Gershmen, Anatole. 423
Ghallab, Malik. 310
Gibbon*, Jeff. 978
Gilreath, Al. 846
Glaser, Frank. 644
Goers. G. 429
Goldin, Sarah E. 212
Goldstein, Ira. 913
Goossens, D. 992
Granger, Richard H. Jr. 354
Greenfeld, Morton R. 978

Haas, Andrew. 382
Hagert. Goran. 178
Hanson, Allen R. 648
Hartley, Roger T. 862
Havens, William S. 625
Hayes, Philip J. 416, 432
Healy. Timothy J. 803
Henschen, Lawrence J. • . . 472, 528
Herold, A. 511
Hinton, Geoffrey E. 683, 1088
Hirschman, Lynette. 289
Hobbs, Jerry. 85, 190
Hollander, Clifford R. 843
Horn, Werner. 850

Iijima, Jun'ichi 779
Ikeuchi, Katsushi. 595
Ishizuka, Mitsuru 837
Israel, David J. 203, 452

Jacobs, Charlotte D. 876
Jacobs, Howard. 343
Jain, Ramesh. 652
Johnson, Paul E. 215
Jones, K. Sparck. . . . .• 443
Joshi, Aravind K. 61, 385
Jouannaud, J.P. 1016

Kahn, Kenneth M. 933
Kaihara, S. 910
Kanade, Takeo. 674, 775
Kanal, Laveen . . . . •. 569
Kanayama, Yutaka •. 779
Kanoui, H. 947, 1056
Kastner, John K. 908
Katz, Shmuel. 1030
Kayaer, Daniel . • •. 64
Kelly, Van. 343

Kennedy, William G. 1065
Kibler, Dennis. 345
King, M. 43
Kirchner, C. 1016
Kirchner, H. 1016
Klahr, Philip. 212
Kline, Paul J. 296
Kodratoff, Yves. 141
Kolbe, Werner. 153
Kolodner, Janet. 227
Konolige, Kurt. 496
Korf, Richard. 1007
Kornfeld, William. 575
Korsin, Martin. 1057
Koyama, T. 910
Krueger, M.W. 362
Kulikowski, Casimir A. 853
Kumar, Vipin. . . . .•. 569
Kurokawa, T. 910

Langley, Pat. 121
Lanka, Sitaram. 61
Latombe, Jean-Claude. 766
Laubsch, Joachim. 964
Lawton, Daryl T. 700
Lebowitz, Michael . . . 13, 348, 1059
Lehnert, Wendy G. 184
Lescanne, Pierre. 548
Lesmo, L. 440
Letsinger, Reed. 829
Levesque, Hector J. 240
Levin, D.Ya. . . . .***
Lieberman, Henry. 1060
Loisel, Regine. 141
London, Philip. 409
Long, James E. 803
Lowe, David. 613
Lowrance, John D. 319
Lucas, Bruce D. 674

MacVicar-Whelan, P.J. 752
Mackworth, Alan K. 625
Maenobu, Kiyoshi. 746
Magnani, D. 440
Marburger, Heinz. 49
Marik, Vladimir. 773
Mark, William. 375
Markuss, Zsussanna. 264
Martin, William A.. 940
Mays, Eric. 61
McAlletter, David A. 1024
McArthur, David. 171, 809
McCarty, L. Thorne. 246
McDermott, John. 824
McDonald, David B. 1061
McDonald, David D. 1062
McGuire, Rod. 58
McKay, Donald. 368
Mero, Laszlo. 572

Michalski, R.S. 460
Minamikawa, T. 910
Mitchell, Tom M. 127, 343
Mittal, S. 1055
Moravec, Hans P. 785
Morris, Paul 345
Mott, David .H. 159

Nagel, Hans-Hellmut 661, 692
Nakano, Hidetoshi 710
Naqvi, Shamim A. 528
Narun'yani, A.S. . . . . ***
Neches, Robert 283
Neumann, Bernd 49, 661
Norman, Donald A. 1097
Novak, Gordon S. Jr. 1063
Novak, Hans-Joachim 49
Nudel, Bernard 127

O'Rourke, Joseph 664, 737
Oakey, S. 109
Ohno, Yutaka 949
Ohta, Yu-ichi 746
Oim, Haldur . . . . ***
Olthoff, Walter 1037
Oshima, Masaki 601
Overton, Kenneth J. 791

Palmer, Martha 277
Papalaskaris, Mary A. 304
Patil, Ramesh S. 893
Pearl, Judea 554
Perkins, W.A. 1066
Prazdny, K. 698
Pruchnik, Paul 846
Purdom, Paul Walton Jr. 588

Radig, Bernd 719
Raulefs, Peter 1037
Reichman, Rachel 19
Reinstein, Harry C. 888
Reiser, Brian J. 209, 184
Reiter, Raymond 270
Rich, Charles 1044
Rieger, Chuck 955, 985
Riesbeck, Christopher K. 115
Riseman, Edward M. 648
Rissland, Edwina L. 162
Rosenberg, R.S. 758
Rosenschein, Stanley J. 331
Rowat, P.F. 758
Rubin, Eric 975
Rubin, Steven 1067

Sabbah, Daniel 607, 722
Sakai, Toshiyuki 746
Sammut, Claude 104
Schooley, Pat 343
Schubert, Lenhart 304
Schwartz, William B. 893
Scott, A. Carlisle 876
Seidel, Raimund 338
Selbig, Joachim 153
Selfridge, Mallory 92, 362
Selfridge, Peter G. 755
Sembugamoorthy, V. 106
Shapiro, Ehud Y. 446, 1064
Shapiro, Stuart C. 368
Shaw, David Elliot 961
Shirai, Yoshiaki 601
Shortliffe, Edward H. 876
Sidner, Candace L. 203
Siekmann, J. 511, 532
Silver, Bernard 551
Simi, Maria 504
Simon, Herbert A. 121
Slagle, James R. 1065
Sleeman, D. H. 882
Sloan, Kenneth R. Jr. . . . 734, 755
Sloman, Aaron 197
Small, Steven 70
Smith, Douglas R. 1027
Smith, J.W. 1055
Smith, Reid G. 343
Smolka, G. 511
Sneiderman, Rich 846
Soga, Itsuya 77
Soloway, Elliot M. 162, 975
Sowizral, Henry 809
Sridharan, N. S. 246
Steele, Barbara 824
Steinacker, Ingeborg 237
Steinberg, Lou 343
Stepp, R. 460
Story, Guy 289
Sugimoto, Shigeo 949
Swartout, William R. 815
Szabo, P. 532
Szolovits, Peter 893, 940

Tabata, Koichi 949
Tarnlund, Sten-Ake 178
Taylor, Gregory B. 388
Teller, Virginia 394
Thompson, William B. 215
Thorndyke, Perry W. 171
Tomita, Fumiaki 728
Torasso, P. 440
Touretzky, David S. 257
Trappl, Robert 850
Trigg, Randy 955
Trost, Harald 237
Tsotsos John 900
Tsuji, Saburo 77, 710

Utgoff, Paul E. 127

van Caneghem, M. 947, 1056
van Melle, William 876
van Roggen, Walter 257
Veroff, Robert L. 472
Villemin, F.Y. 1004

Walter, Ch. 511
Waltz, David L. 1
Waters, Richard C. 920
Webb, Jon A. 686
Webber, Bonnie L. 61
Weiner, James L. 277
Weinstein, Scott 385
Weir, Sylvia 970
Weiss, Sholom M. 853, 908
Wesley, Leonard P. 144
Weymouth, Terry E. 628
Whitehill, Stephen B. 388
Wilczynski, David 135
Wilensky, Robert 25, 930
Williams, Thomas 791
Wolf, Thomas C. 1057
Wong, Douglas 7
Wood, Richard 985
Woolf, Beverly 975
Wysotzki, Fritz 153

Yachida, Masahiko 716
Yang, CJ 47
Yao, James T.P. 837
Yonke, Martin D. 978
York, Bryant W. 648
Yuta, Shin'ichi 779

Zarri, Gian Piero 401
Zdrahal, Zdenek 680
Zdybel, Frank 978
Zimmerman, Ruth 1030
Zucker, Steven 1102