Ordering techniques for local elimination algorithms*

O. A. Shcherbina*, A. V. Sviridenko†

*University of Vienna, oleg.shcherbina@univie.ac.at
†Tavrian National University, oleks.sviridenko@gmail.com

1 Introduction

The use of discrete optimization (DO) models and algorithms makes it possible to solve many real-life problems in scheduling theory, optimization on networks, routing in communication networks, facility location in enterprize resource planing, and logistics. Applications of DO in the artificial intelligence field include theorem proving, SAT in propositional logic, robotics problems, inference calculation in Bayesian networks, scheduling, and others.

Many real-life discrete optimization problems (DOPs) contain a huge number of variables and/or constraints that make the models intractable for currently available DO solvers. Usually, such problems have a special structure, and the matrices of constraints for large-scale problems are sparse. The nonzero elements of the matrices often involve a limited number of blocks. The block form of many DO problems is usually caused by the weak connectedness of subsystems of real-world systems.

Graph-based local elimination algorithms [8] that include nonserial dynamic programming (NSDP) [2, 8], its modifications, and tree decomposition combined with dynamic programming are promising decomposition approaches that allow exploiting the structure of discrete problems. Aforementioned methods use just the local information (i.e., information about elements of given element's neighborhood) in a process of solving discrete problems [11]. NSDP (or Variable Elimination Algorithm) eliminates variables of DOP using an elimination order which makes significant impact on running time. As finding an optimal ordering is NP-complete [10], heuristics are utilized in practice for finding elimination orderings. The literature has reported extensive computational results for the use of different ordering heuristics in the solution of systems of equations [1], [3]. However, no such experiments have been reported for NSDP to date.

Given the increased recent interest in DOPs, the subject of experimental research of NSDP algorithms that utilize heuristic variable orderings is timely. In this paper, we present comparative computational results from the benchmarking of five ordering techniques, namely: minimum degree ordering, nested dissection ordering, maximum cardinality search, minimum fill-in, and lexicographic breadth-first search.

2 Variable Elimination Algorithms

Consider a DOP with constraints:

$$F(x_1, x_2, \ldots, x_n) = \sum_{k \in K} f_k(Y^k) \to \max \qquad (1)$$

subject to the constraints

$$g_i(X_{S_i})\ R_i\ 0, \quad i \in M = \{1, 2, \ldots, m\}, \qquad (2)$$

$$x_j \in D_j, \quad j \in N = \{1, \ldots, n\}, \qquad (3)$$

where
$X = \{x_1, \ldots, x_n\}$ is a set of discrete variables, $Y^k \subseteq \{x_1, x_2, \ldots, x_n\}, k \in K = \{1, 2, \ldots, t\}$, t – number of components of objective function, $S_i \subseteq \{1, 2, \ldots, n\}$, $R_i \in \{\leq, =, \geq\}, i \in M$; D_j

*This research is partly supported by FWF (Austrian Science Funds) under the project P20900-N13.

is a finite set of admissible values of variable x_j, $j \in N$. The functions $f_k(X^k)$, $k \in K$ are called components of the objective function and can be defined in tabular form. We use here a notation: if $S = \{j_1, \ldots, j_q\}$ then $X_S = \{x_{j_1}, \ldots, x_{j_q}\}$.

Consider a sparse discrete optimization problem (1) – (3) whose structure is described by an undirected interaction graph $G = (X, E)$. Solve this problem with a NSDP.
Given ordering of variable indices α the NSDP proceeds in the following way: it subsequently eliminates $x_{\alpha_1}, \ldots, x_{\alpha_n}$ in the current graph and computes an associated local information $h_i(Nb(x_{\alpha_i}))$ about vertices from $Nb(x_{\alpha_i})$ $(i = 1, \ldots, n)$. This can be described by the combinatorial elimination process:

$$G^0 = G, G^1, \ldots, G^{j-1}, G^j, \ldots, G^n,$$

where G^j is the x_{α_j}-elimination graph of G^{j-1} and $G^n = \emptyset$

3 Elimination Ordering Techniques

An efficiency of the NSDP crucially depends on the interaction graph structure of a DOP. If the interaction graph is rather sparse or, in other words, has a relatively small induced width, then the complexity of the algorithm is reasonable. At the same time an interaction graph leads us to another critical factor such as an elimination order which should be obtained from the interaction graph.
From the other side the NSDP heavily depends on the elimination ordering. A good elimination ordering yields small cliques during variable elimination. There are several successful schemes for finding a good ordering which we will used in this paper: *minimum degree ordering algorithm* (MD), *nested dissection ordering algorithm* (ND), *maximum cardinality search algorithm* (MCS), *minimum fill-in heuristic* (MIN-FILL) and *lexicographic breath-first search algorithm* (LEX-BFS).

3.1 Minimum degree ordering algorithm

The minimum degree (MD) ordering algorithm [1] is one of the most widely used in linear algebra heuristic, since it produces factors with relatively low fill-in on a wide range of matrices.
In the minimum degree heuristic, a vertex v of minimum degree is chosen. The graph G', obtained by making the neighborhood of v a clique and then removing v and its incident edges, is built. Recursively, a chordal supergraph H' of G' is made with the heuristic. Then a chordal supergraph H of G is obtained, by adding v and its incident edges from G to H'. To create an elimination order with help of minimum degree ordering algorithm the *minimum_degree_ordering*() function from BOOST library (http://www.boost.org) has been used.

3.2 Nested dissection algorithm

To create an elimination order, we recursively partition the elimination graph using nested dissection. More specifically, we use $METIS\_EdgeND()$ function from METIS library [5] to find a nested dissection ordering.

3.3 Maximum cardinality search algorithm

The Maximum Cardinality Search (MCS) algorithm [9] visits the vertices of a graph in an order such that at any point, a vertex is visited that has the largest number of visited neighbors. An MCS-ordering of a graph is an ordering of the vertices that can be generated by the Maximum Cardinality Search algorithm. The visited degree of a vertex v in an MCS-ordering is the number of neighbors of v that are before v in the ordering.
To create an elimination ordering the *chompack.maxchardsearch*() function from the **Chordal Matrix Package** (CHOMPACK) (http://abel.ee.ucla.edu/chompack/) has been used.

3.4 Minimum Fill-in algorithm

The minimum fill-in heuristic [4] works similarly with minimum degree heuristic, but now the vertex v is selected such that the number of edges that is added to make a neighborhood of v a clique is as small as possible.

3.5 Lexicographic breadth-first search algorithm

Lexicographic breadth-first search algorithm (LEX-BFS) [7] numbers the vertices from n to 1 in the order that they are selected. This numbering fixes the positions of an elimination scheme. For each vertex v, the label of v will consist of a set of numbers listed in decreasing order. The vertices can then be lexicographically ordered according to their labels.

4 Benchmarking

4.1 NSDP algorithm implementation

The NSDP algorithm was implemented by the first author in Python. The ND and MD algorithms were implemented in C and C++, respectively.

4.2 Test problems

For benchmarking the DO test problems were generated by using hypergraphs from the CSP[1] hypergraph library [6]. This collection contains various classes of constraint hypergraphs from industry (DaimlerChrysler, NASA, ISCAS) as well as synthetically generated ones (e.g. Grid or Cliques).
The test problems were generated in the following way. The constraints structure of a linear DO problem with binary variables was described by hypergraph from the library [6]. To build constraint i the next hyperedge of hypergraph was taken, which includes a set of variables X_{S_i} for a new building constraint. In the next step, the coefficients for appropriate variables of A_{S_i} were generated using a random number generator. Then the left part of i-th constraint had view $A_{S_i}X_{S_i}$, while the right part was $\sigma \sum A_{S_i}$, where σ is random number from interval (0, 1). Objective function is linear and includes all variables – vertices of hypergraph, where coefficients c_j of objective function $\sum_{j=1}^{n} c_j x_j \rightarrow \max$ where created with help of random number generator.
After the test problems were generated, the ordering algorithms MD, ND, MCS, MIN-FILL and LEX-BFS were applied for obtaining an elimination ordering. Then the problems were solved with the NSDP algorithm by utilizing to the specified elimination ordering.

[1]CSP – Constraint Satisfaction Problem.

4.3 Benchmarking ordering analysis

The following five groups of 33 test problems have been taken: 'dubois', 'bridge', 'adder', 'pret' and 'NewSystem'. All experimental results were obtained on a machine with Intel Core 2 Duo processor 2.66 GHz, 2 GB main memory and operating system Linux, version 2.6.35-24-generic. We can see that for ND algorithm the minimal run-time of the NSDP was achieved 0 times (0 %), for MD 2 times (6,0 %), LEX-BFS 3 times (9,1 %), MCS 9 times (27,3 %) and MIN-FILL 19 times (57,6 %).

Let us take a look at the results in more detail. The benchmarking results for the groups of test problems 'dubois', 'bridge', and 'adder' show to us that MCS, MIN-FILL and LEX-BFS heuristics behave quite similar and give the best result for a given group of problems. At the same time MD and ND show the worst time result. Also for the group 'bridge' we can see the gradual decreasing of the time result for the LEX-BFS heuristics and the obvious domination of MIN-FILL.
Here we can see the importance of the right choice of heuristics for a certain group of problems. In the case of 'pret', we see obvious domination of MIN-FILL algorithm, while MCS and LEX-BFS fall behind. However, the group 'NewSystem' shows completely opposite results, where MIN-FILL runs third while MCS and LEX-BFS take the first two places.
In the case of 'pret', we see the obvious domination of the MIN-FILL algorithm, while MCS and LEX-

BFS go back. But the group 'NewSystem' shows the completely opposite result, where MIN-FILL goes on the third place while MCS and LEX-BFS take the first places.

5 Conclusion

The goal of this paper to research the role of five variable ordering algorithms and to describe the effect that they play on solving time of sparse DO problems with help of the NSDP. Our computational experiments demonstrate that, for solving DO problems, variable ordering has a significant impact on the run-time for solving the problem. Furthermore, different ordering heuristics were observed to be more effective for different classes of problems. Overall, the MCS and MIN-FILL heuristics have provided the best results for solving DO problems of the problem classes that were considered in this paper. It seems promising to continue this line of research by studying methods of block elimination with suitable partitioning methods.

References

[1] P. R. Amestoy , T. A. Davis and I. S. Duff, An approximate minimum degree ordering algorithm, *SIAM Journal on Matrix Analysis and Applications*, V. 17, N 4, 886-905, 1996.

[2] U. Bertele and F. Brioschi, *Nonserial Dynamic Programming*, Academic Press, 1972.

[3] J. A. George, Nested dissection of a regular finite element mesh, *SIAM J Numer Anal*, V. 10, 345-367, 1973.

[4] P. Jégou, S. N. Ndiaye and C. Terrioux, Computing and exploiting tree-decompositions for (Max-)CSP, *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, P. 777-781, 2005.

[5] G. Karypis and V. Kumar, MeTiS a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Version 4. University of Minnesota, 1998.

[6] N. Musliu, M. Samer, T. Ganzow and G. Gottlob, A csp hypergraph library, Technical Report, DBAI-TR-2005-50, Technische Universität Wien, 2005.

[7] D. J. Rose, R. Tarjan and G. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.*, V. 5, 266-283, 1976.

[8] O. Shcherbina Graph-Based Local Elimination Algorithms in Discrete Optimization. In: *Foundations of Computational Intelligence Volume 3. Global Optimization Series: Studies in Computational Intelligence*, Vol. 203 / Abraham A, Hassanien A-E, Siarry P, Engelbrecht A (Eds). Springer Berlin / Heidelberg, 2009, P. 235–266.

[9] R. E. Tarjan and M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.*, V. 13, 566-579, 1984.

[10] M. Yannakakis, Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, V. 2, 77-79, 1981.

[11] Y. I. Zhuravlev, Local algorithm of information computation (Russian), I,II. *Kibernetika* 1965, Vol. 1, pp. 12-19; 1966, Vol. 2, pp. 1-11.