1. Introduction

The use of discrete optimization (DO) models and algorithms makes it possible to solve many practical problems, since the discrete optimization models correctly represent the nonlinear dependence, indivisibility of an objects, consider the limitations of logical type and all sorts of technology requirements, including those that have qualitative character. To meet the challenge of solving large scale DO problems (DOPs) in reasonable time, there is an urgent need to develop and study new decomposition approaches. Among decomposition approaches appropriate for solving sparse DO problems we mention local elimination algorithms (LEA) [1], [2], which can exploit sparsity in the interaction graph of a DOP and allow to compute a solution in stages such that each of them uses results from previous stages. LEAs compute global information using local computations (i.e., computations of information about elements of neighborhoods of variables or constraints - usually, solving subproblems).

The algorithmic scheme of LEA is defined by an elimination tree [3] whose vertices are associated with subproblems and whose edges express information on interdependence between subproblems. The structure of a DO problem can be defined either by an interaction graph of initial elements (variables and constraints), or by various derived structures, e.g., block structures, block-tree structures defined by a so called condensed graph.

There are various computational schemes for realizing LEA, including the LEA elimination of variables, block-elimination algorithm, LEA based on tree decomposition.

2. Block local elimination scheme

2.1. Partitions, clustering, and quotient graphs

Consider the integer linear programming (ILP) problem with binary variables
\[ f(X) = CX = \sum_{j=1}^{n} c_j x_j \rightarrow \text{max} \]

subject to constraints
\[
\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m, \\
x_j = 0, 1, \quad j = 1, 2, \ldots, n.
\]

The local elimination procedure can be applied to elimination of not only separate variables but also to sets of variables and can use the so-called elimination of variables in blocks, which allows to eliminate several variables in block.

Applying the method of merging variables into meta-variables allows to obtain condensed or meta-DOPs which have a simpler structure. If the resulting meta-DOP has a nice structure (e.g., a tree structure) then it can be solved efficiently.

An ordered partition of a set \( X \) is a decomposition of \( X \) into ordered sequence of pairwise disjoint nonempty subsets whose union is all of \( X \).

In general, graph partitioning is \( \mathcal{NP} \)-hard. Since graph partitioning is difficult in general, there is a need for approximation algorithms. A popular algorithm in this respect is \textbf{MeTiS}\(^1\), which has a good implementation available in the public domain.

An important special case of partitions are so-called blocks. Two variables are indistinguishable if they have the same closed neighborhood. A block is a maximal set of indistinguishable vertices. The blocks of \( G \) partition \( X \) since indistinguishability is an equivalence relation defined on the original vertices. The corresponding graph is called condensed graph, which is a merged form of original graph.

An equivalence relation on a set induces a partition on it, and also any partition induces an equivalence relation. Given a graph \( G = (X, E) \), let \( X \) be a partition on the vertex set \( X: X = x_1, x_2, \ldots, x_p, \quad p \leq n \), where \( x_l = X_{K_l} (K_l \text{ is a set of indices corresponding to } x_l, \quad l = 1, \ldots, p) \). For this ordered partition \( X \), the DOP (1) – (3) can be solved by the LEA using quotient interaction graph \( G \).

\(^1\)http://www-users.cs.umn.edu/~karypis/metis
That is, \( \bigcup_{i=1}^{p} x_i = X \) and \( x_i \cap x_k = \emptyset \) for \( i \neq k \). We define the quotient graph of \( G \) with respect to the partition \( X \) to be the graph
\[
G = G/X = (X, E),
\]
where \( (x_i, x_k) \in E \) if and only if \( Nb_G(x_i) \cap x_k \neq \emptyset \).

The quotient graph \( G(X, E) \) is an equivalent representation of the interaction graph \( G(X, E) \), where \( X \) is a set of blocks (or indistinguishable sets of vertices), and \( E \subseteq X \times X \) be the edges defined on \( X \). A local block elimination scheme is one in which the vertices of each block are eliminated contiguously. As an application of a clustering technique we consider below a block local elimination procedure where the elimination of the block (i.e., a subset of variables) can be seen as the merging of its variables into a meta-variable.

2.2. Block local elimination algorithm

A. Forward part

Consider first the block \( x_1 \). Then
\[
\max_{X} \{ CN_X | A_{iS_i} X_{S_i} \leq b_i, \ i \in M, \ x_j = 0,1, \ j \in N \} =
\max_{X_{K_1}} \{ CN_{-K_1} X_{N-K_1} + h_1(Nb(X_{K_1})) | A_{iS_i} X_{S_i} \leq b_i, \ i \in M - U_1, \ x_j = 0,1, \ j \in N - K_1 \}
\]
where \( U_1 = \{ i : S_i \cap K_1 \neq \emptyset \} \) and
\[
h_1(Nb(X_{K_1})) = \max_{X_{K_1}} \{ C_{K_1} X_{K_1} | A_{iS_i} X_{S_i} \leq b_i, \ i \in U_1, \ x_j = 0,1, \ x_j \in Nb(x_1) \}.
\]
The first step of the local block elimination procedure consists of solving, using complete enumeration of \( X_{K_1} \), the following optimization problem
\[
h_1(Nb(X_{K_1})) = \max_{X_{K_1}} \{ C_{K_1} X_{K_1} | A_{iS_i} X_{S_i} \leq b_i, \ i \in U_1, \ x_j = 0,1, \ x_j \in Nb(x_1) \},
\]
and storing the optimal local solutions \( X_{K_1} \) as a function of the neighborhood of \( X_{K_1} \), i.e., \( X_{K_1}(Nb(X_{K_1})) \).

The maximization of \( f(X) \) over all feasible assignments \( Nb(X_{K_1}) \), is called the elimination of the block (or meta-variable) \( X_{K_1} \). The optimization problem left after the elimination of \( X_{K_1} \) is:
\[
\max_{X_{-K_1}} \{ CN_{-K_1} X_{N-K_1} + h_1(Nb(X_{K_1})) | A_{iS_i} X_{S_i} \leq b_i, \ i \in M - U_1, \ x_j = 0,1, \ j \in N - K_1 \},
\]
Note that it has the same form as the original problem, and the tabular function $h_1(Nb(X_{K_1}))$ may be considered as a new component of the modified objective function. Subsequently, the same procedure may be applied to the elimination of the blocks – meta-variables $x_2 = X_{K_2}, \ldots , x_p = X_{K_p}$, in turn. At each step $j$ the new component $h_{x_j}$ and optimal local solutions $X_{K_j}$ are stored as functions of $Nb(X_{K_j} | X_{K_1}, \ldots , X_{K_{j-1}})$, i.e., the set of variables interacting with at least one variable of $X_{K_j}$ in the current problem, obtained from the original problem by the elimination of $X_{K_1}, \ldots , X_{K_{j-1}}$. Since the set $Nb(X_{K_p} | X_{K_1}, \ldots , X_{K_{p-1}})$ is empty, the elimination of $X_{K_p}$ yields the optimal value of objective $f(X)$.

### B. Backward part.

This part of the procedure consists of the consecutive choice of $X_{K_p}, X_{K_{p-1}}, \ldots , X_{K_1}$, i.e., the optimal local solutions from the stored tables $X_{K_1}^*(Nb(X_{K_1})), X_{K_2}^*(Nb(X_{K_2} | X_{K_1})), \ldots , X_{K_p}^* | X_{K_{p-1}}, \ldots , X_{K_1}$.

Underlying DAG of the local block elimination procedure contains nodes corresponding to computing of functions $h_{x_i}(Nb_{G_{i-1}}(X_i))$ and is a generalized elimination tree.

### 3. Comparative computational experiment

Among extremely important research questions about the effectiveness of local elimination algorithms (LEA), the next one causes special interest: “Is the use of LEA in combination with a discrete optimization (DO) algorithm (for solving problems in the blocks) consistently more efficient than the standalone use of the DO algorithm?” [4].

The computational capabilities of the LEA in combination with a modern solver were tested by using SYMPHONY\(^2\) as the implementation framework. SYMPHONY is part of the COIN-OR\(^3\) project and it can solve mixed-integer linear programs (MILP) sequentially or in parallel. We chose this framework since it is open-source and supports warm restarts, which implement postoptimal analysis (PA) of ILP problems.

All experimental results were obtained on an Intel Core 2 Duo at 2.66 GHz machine with 2 GB main memory, and running Linux, version 2.6.35-24-generic. SYMPHONY 5.4.1\(^4\) was used for the LEA implementation. The maximum solving time is denoted by $TIMEOUT$, and is equal to 2

---

\(^2\)https://projects.coin-or.org/SYMPHONY

\(^3\)http://www.coin-or.org

\(^4\)http://www.coin-or.org/download/source/SYMPHONY/
hours. All the ILP problems with binary variables from a given experiment have artificially generated quasi-block structures. All the blocks from a single problem have the same number of variables, and also the same number of variables in separators between them. This is required in order to evaluate the impact of the PA on the time to solve the problem by increasing the number of variables.

The test problems were generated by specifying the number of variables, the number of constraints and the size of the separators between blocks. The objective function and constraint matrix coefficients, and the right-hand sides for each of the block were generated by using a pseudorandom-number generator.

Each test problem was solved by using three algorithms, a) the basic MILP SYMPHONY solver with the OsiSym interface, b) the LEA in combination with SYMPHONY, c) the LEA in combination with SYMPHONY and with PA (warm restarts). In all the cases SYMPHONY used preprocessing.

The computational experiments are described in details in [5] and show that LEA combined with SYMPHONY for solving quasi-block problems with small separators outperforms the stand alone SYMPHONY solver. Additionally, by increasing the size of the separators in the problems for the same number of variables and block sizes LEA becomes less efficient due to the increased number of iteration for solving the block subproblems.

REFERENCES