Local elimination algorithms in discrete optimization

O. A. Shcherbina*

*University of Vienna, oleg.shcherbina@univie.ac.at

The use of discrete optimization (DO) models and algorithms makes it possible to solve many practical problems in scheduling theory, network optimization, routing in communication networks, facility location, optimization in enterprise resource planning, and logistics.

Usually, discrete optimization problems (DOPs) from applications have a special structure, and the matrices of constraints for large-scale problems have a lot of zero elements (sparse matrices). Graph-based structural decomposition techniques, e.g., nonserial dynamic programming (NSDP) (Bertele, Brioschi [1], Shcherbina [3]), its modifications, and tree decomposition combined with dynamic programming [4] are promising decomposition approaches that allow exploiting the structure of discrete problems.

It is important that aforementioned methods use just the local information (i.e., information about elements of given element's neighborhood) in a process of solving discrete problems [6]. A class of local elimination algorithms is proposed in [5] as a general framework that allows to calculate some **global information** about a solution of the entire problem using **local computations**.

The structure of discrete optimization problems is described by the so called **structural graph**. The **structural graph** can be the interaction graph of the original elements (for example, between the variables of the problem) or the **quotient graph**. The quotient graph can be obtained by merging a set of original elements (for example, a subgraph) into a condensed element. The original subset (subgraph) that formed the condensed element is called the **detailed graph** of this element.

A LEA [5] eliminates local elements of the problem's structure defined by the structural graph by computing and storing local information about these elements in the form of new dependencies added to the problem. Thus, the local elimination procedure consists of two parts:

A. The **forward part** eliminates elements, computes and stores local solutions, and finally computes the value of the objective function;

B. The **backward part** finds the global solution of the whole problem using the tables of local solutions; the global solution gives the optimal value of the objective function found while performing the forward part of the procedure.

The LEA analyzes a **neighborhood** $Nb(x)$ of the current element x in the structural graph of the problem, applies an **elimination operator** (which depends on the particular problem) to that element, calculates the function $h(Nb(x))$ that contains **local information** about x, and finds the local solution $x^*(Nb(x))$. Next, the element x is eliminated, and a clique is created from the elements of $Nb(x)$. The elimination of elements and the creation of cliques changes the structural graph and the neighborhoods of elements. The backward part of the local elimination algorithm reconstructs the solution of the whole problem based on the local solutions $x^*(Nb(x))$.

The algorithmic scheme of the LEA is a **DAG** in which the vertices correspond to the local subproblems and the edges reflect the informational dependence of the subproblems on each other.

Consider a sparse DOP in the following form

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

subject to

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

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

where $X = \{x_1, \ldots, x_n\}$ is a set of discrete variables, $X^k \subseteq \{x_1, x_2, \ldots, x_n\}, k \in K = \{1, 2, \ldots, t\}$, t – number of components in the objective function, $S_i \subseteq \{1, 2, \ldots, n\}$, $R_i \in \{\leq, =, \geq\}, i \in M$; D_j is a finite set of admissible values of variable x_j, $j \in N$. 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 the notation: if $S = \{j_1, \ldots, j_q\}$ then $X_S = \{x_{j_1}, \ldots, x_{j_q}\}$.

Introduce a graph representation of the DOP. The structural graph of the DOP defines which variables are in which constraints. Structure of a DOP can be defined either by interaction graph of initial elements (variables in the DOP) or by various derived structures, e.g., block structures, block-tree structures defined by the **quotient** graph.

Concrete choice of a structural graph of the DOP defines different local elimination schemes: nonserial dynamic programming, block decomposition, tree decomposition etc.

Variables $x \in X$ and $y \in X$ **interact** in the DOP with constraints if they both appear either in the same component of objective function, or in the same constraint (in other words, if variables are both either in a set X^k, or in a set X_{S_i}). An **interaction graph** [1] represents a structure of the DOP in a natural way. **Interaction graph** of the DOP is an undirected graph $G = (X, E)$, such that 1) vertices X of G correspond to variables of the DOP; 2) two vertices of G are adjacent iff corresponding variables interact.

Set of variables interacting with a variable $x \in X$ is denoted by $Nb(x)$ and called the **neighborhood** of the variable x. For corresponding vertices a neighborhood of a vertex x is a set of vertices of interaction graph that are linked by edges with x. Denote the latter neighborhood as $Nb_G(x)$.

If the DOP is divided into blocks corresponding to subsets of variables (meta-variables) or to subsets of constraints (meta-constraints), then block structure can be described by a **structural quotient (condensed) graph**, whose meta-nodes correspond to subsets of the variables of blocks and meta-edges correspond to adjacent blocks.

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 local elimination algorithm (LEA). LEA uses an ordering α of X: Given a graph $G = (X, E)$ an **ordering** α of X is a bijection $\alpha : X \leftrightarrow \{1, 2, \ldots, n\}$ where $n = |X|$. G_α and X_α are correspondingly an ordered graph and an ordered vertex set.

In G_α, a **monotone neighborhood** $\overline{Nb}^{\alpha}_{G}(x_i)$ of $x_i \in X$ is a set of vertices **monotonely adjacent** to a vertex x_i, i.e. $\overline{Nb}^{\alpha}_{G}(x_i) = \{x_j \in Nb_G(x_i) | j > i\}$. The graph G_x obtained from $G = (X, E)$ (i) adding edges so that all vertices in $Nb_G(x)$ are pairwise adjacent, and (ii) deleting x and its incident edges is the x–**elimination** graph of G. This process is called the **elimination** of the vertex x.

Let us introduce the notion for the **elimination tree** (etree) [2]. Given a graph $G = (X, E)$ and an ordering α, the **elimination tree** is a directed tree $\overrightarrow{T}_\alpha$ that has the same vertices X as G and its edges are determined by a parent relation defined as follows: the parent x is the **first vertex** (according to the ordering α) of the monotone neighborhood $\overline{Nb}^{\alpha}_{G^+_\alpha}(x)$ of x in the filled graph G^+_α.

Using the parent relation introduced above we can define a directed filled graph $\overrightarrow{G}^+_\alpha$.

The underlying DAG of a local variable elimination scheme can be constructed in the following way. At step i, we represent the computation of the function $h_{x_i}(Nb_{G_{x_{i-1}}}(x_i))$ as a node of the DAG (corresponding to the vertex x_i). Then, this node containing variables $(x_i, Nb^{(i-1)}_{G_{x_{i-1}}}(x_i))$ is linked with a first x_j (accordingly to the ordering α) which is in $Nb_{G^{(i-1)}_{x_{i-1}}}(x_i)$.

It is easy to see that the elimination tree is the DAG of the computational procedure of the LEA. Using the elimination tree it is possible to build a corresponding tree decomposition.

References

[1] U. Bertele and F. Brioschi, *Nonserial dynamic programming.* Academic Press, 1972.

[2] J. W. H. Liu, The role of elimination trees in sparse factorization, *SIAM Journal on Matrix Analysis and Applications*, 1990, Vol. 11, pp. 134-172.

[3] O. Shcherbina, Nonserial dynamic programming and tree decomposition in discrete optimization, *Proc. of Int. Conference on Operations Research "Operations Research 2006"*, Karlsruhe, 6-8 September, 2006, Springer Verlag, pp. 155-160, 2007.

[4] O. A. Shcherbina, Tree decomposition and discrete optimization problems: A survey, *Cybernetics and Systems Analysis*, 2007, Vol. 43, pp. 549-562.

[5] O. A. Shcherbina, Local elimination algorithms for solving sparse discrete problems, *Computational Mathematics and Mathematical Physics*, 2008, Vol. 48, pp. 152-167.

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