Aplikace matematiky

Jaroslav Hrouda

Staging in Balas' algorithm

Aplikace matematiky, Vol. 16 (1971), No. 5, 354–369

Persistent URL: http://dml.cz/dmlcz/103367

STAGING IN BALAS' ALGORITHM

JAROSLAV HROUDA

(Received June 26, 1970)

INTRODUCTION

In this article the following idea is developed: to break up an enumeration process into parts, not horizontally like a decomposition, but vertically in a sense of stratifying the enumeration process. Such partitioning (staging) arises: (1) when one attempts to bound by trial the minimal value of the objective function in a problem of zero-one integer programming, (1a) when all suboptimal solutions in a given zone of values of the objective function are wanted, (2) when the right-hand sides of the problem depend linearly on a discrete parameter.

Our approach to the solution of such partitioned problems can be characterized as an extended use of Glover's bookkeeping of the enumeration process B (see [1]). This is now recorded in a "long" sequence containing, in addition, indications about process interruptions caused by staging. The sequence currently discards the elements needless for subsequent stages. Thus we may interpret this idea as a recording methodology – the staged enumeration process BG – which allows of more general use than a simple enumeration process does (particularly, it makes possible a parameterization in the zero-one integer programming problem).

Now we briefly mention the contents of the individual paragraphs. In § 1 the principle of the staged enumeration process BG (abbreviated as SBG) is explained, its motivation given, and a terminology built up. Then some special problems are formulated corresponding to (1), (1a), and (2) above. In § 2 an algorithm realizing the process SBG is described (referred to as Algorithm SBG) and its adaptations for the solution of the special problems are given. It fairly exploits the components of Algorithm BG from [1]. In § 3 Algorithm SBG is justified by reducing it to the conceptually and mathematically simpler Algorithm BG. In § 4 some recommendations are given concerning the realization of Algorithm SBG in a computer.

Notice. This article very closely relates to the previous paper [1] whose knowledge (at least §§ 1 and 2) is therefore urgently demanded.

§ 1. ENUMERATION PROCESS SBG

We shall use the notation $\mathcal{M} = \{0, 1, \ldots, m\}$, $\mathcal{N} = \{1, \ldots, n\}$ and a term' zero-one vector' for a vector with components zero or one. The basic task before us is to solve

Problem T. *Find all zero-one vectors* $x = (x_1, \ldots, x_n)$ *satisfying*

$$\sum_{j=1}^{n} a_{ij}x_j \leqq b_i + td_i \quad (d_i \geqq 0,\ i \in \mathcal{M}) \tag{1.1}$$

successively for $t = t_1 < t_2 < \ldots < t_g$.

We shall call these vectors *t-feasible solutions*. The corresponding $\{j \in \mathcal{N} \mid x_j = 1\}$ are t-feasible combinations. The k-th part of Problem T, i.e. the determination of all t_k-feasible solutions, will be referred to as Problem $T^{(k)}$; its solution constitutes the *k-th stage* in the solution of T.

To solve Problem T, we may use the enumeration process B (Process B) at least in two ways:

(1) Solve Problems $T^{(1)}, T^{(2)}, \ldots$ successively as Problems U.

(2) Make use of the fact that each t_{k-1}-feasible combination is at the same time t_k-feasible, and no t_k-infeasible combination is t_{k-1}-feasible $(1 < k \leqq g)$. Then solve Problem $T^{(g)}$ examining each t_g-feasible combination on its $t_{g-1}, t_{g-2}, \ldots$ up to possibly t_1-feasibility.

The way (1) seems, at a glance, to be uneconomical one. But it could be profitable when the number of stages really performed is less than the number initially intended (owing to lack of a priori information). It surpasses the way (2) also in leaving more freedom when the problem is being stated — the values of parameter t may be specified even during the computation (with regard to intermediate results).

We are going to try to increase efficiency of the way (1) by modifying Process B — in its form BG — so as to make use, in each stage, of an information about the process from the preceding stages. An economical coding of this information and a suitable linking of separate stages is the purpose of this modification, which we shall call *staged enumeration process* BG, or briefly, Process SBG. Here is an outline of it.

We distinguish two types of DBL (at the k-th stage of the process):

(β_1) CDBL (closed DBL): further IBL is impossible or immaterial with respect to solving Problems $T^{(h)}$, $h \geqq k$.

(β_2) ODBL (open DBL): further IBL is impossible or immaterial with respect to solving Problem $T^{(k)}$, but not $T^{(h)}$, $h > k$.

Particularly it follows that for $T^{(g)}$ only CDBL can take place. In correspondence to the two types of DBL we may speak about *closed* or *open branches*; the distinction between them consists in whether during their creation none or at least one ODBL appeared.

The run of the enumeration process, considered as a sequence of operations IBL and DBL, is recorded by means of a sequence $\Phi = \{\varphi_r\}$ the elements of which have the following meaning:

$0 < \varphi_r \leqq n$: add an element φ_r to a combination;

$-n \leqq \varphi_r < 0$: drop out $|\varphi_r|$ elements which were added last to a combination.

The part of the sequence Φ corresponding to a branch (including appropriate "back" elements $\varphi_r < 0$) is called a *loop*. This may be *closed* or *open* according to whether the corresponding branch is closed or open. A pair of elements $(\varphi_r, \varphi_{r+1} < 0)$ where φ_{r+1} was produced by ODBL is a *gap*. An open loop contains at least one gap, a closed one none.

Let us assume that Problems $T^{(1)}, \ldots, T^{(k-1)}$ $(1 < k \leqq g)$ have been solved and a sequence $\Phi^{(k-1)}$ registering the last executed enumeration process is available. Now Problem $T^{(k)}$ is to be solved. Obviously, it is sufficient to find only "new" t_k-feasible solutions (which are not t_{k-1}-feasible); then these will be added to the feasible solutions available from the preceding stages so that we may obtain a complete set of feasible solutions of $T^{(k)}$. For the sake of brevity, we shall use the term *t_{q-p}-feasible solution* $(q > p)$ for a t_q-feasible solution which is not t_p-feasible.

For the solution of Problem $T^{(k)}$ it is necessary that all open branches of the preceding process should be "extended" as far as the constraints of $T^{(k)}$ permit, i.e. as long loops as possible should be built upon all gaps of the sequence $\Phi^{(k-1)}$. We shall reach the gaps successively, forming a sequence Γ by means of elements from $\Phi^{(k-1)}$ and examining combinations thus formed on their $t_{k-(k-1)}$-feasibility. So when "passing through" $\Phi^{(k-1)}$ we avoid any testing as well as selecting elements for IBL. Moreover, we may omit all closed loops unless they contain a $t_{g-(k-1)}$-feasible combination; such loops are called *removable* (so are the corresponding branches).

Simply, we eliminate each removable loop as early as it appears. However, it is desirable that the process should remain redundant even with regard to the eliminated loops (branches). For this purpose we "remember" in Φ each eliminated loop by an element $\tilde{\varphi}_r = n + \varphi_r$ where φ_r is the first element of the loop (the outset of the branch). These elements then participate in the construction of the sequence Γ. Finally, the fact that a loop contains a $t_{g-(k-1)}$-feasible combination is coded by inserting an element $\varphi_r = 0$ at a point where the feasibility has appeared.

From this outline it may now be obvious what we consider to be the advantage of Process SBG: That part of it which reproduces the results of preceding stages by means of the sequence Φ will be quite economical computationally – it requires minimum of vector operations (testing and selecting for IBL being avoided) and the reproduction of preceding stages is performed in a shorter way due to the omission of removable closed loops. This is an optimistic view, of course. A pessimist, on the other hand, will rather emphasize how the preparation of the sequence Φ as well as its analysis and storing in an external memory of a computer is laborious. The final effect will certainly depend on the nature of the problem solved and the tests used,

i.e. on the efficiency of the tests in the realization of ODBL: whether it is true that $t_{k-(k-1)}$-feasible solutions are mostly or completely obtained just in the k-th stage (not sooner which would require their recording into Φ); we call such process *well* or *perfectly stratified*, respectively.

Now we shall formulate some special cases of Problem T which are of practical importance. In all of them a particular role is played by the 0-th function in (1.1), the so called *objective function*, which is to be minimized. We use in this connexion a common term *optimal solution*. Let us denote $\mathcal{M}_t = \{i \in \mathcal{M} \mid d_i > 0\}$, $z(x) = \sum_{j=1}^{n} a_{0j}x_j$, $w = \sum_{j=1}^{n} \max\{a_{0j}, 0\}$, $v = w - \sum_{j=1}^{n} |a_{0j}|$.

Problem T1. *Setting*

$$z_* = \begin{cases} \min\limits_{x} \{z(x) \mid \sum\limits_{j=1}^{n} a_{ij}x_j \leqq b_i \quad (i \in \mathcal{M} - \{0\}), \quad x_j = 0 \text{ or } 1\} \\ w \quad \textit{if the minimum does not exist} \end{cases}$$

and limiting Problem T *by the conditions*

$$\mathcal{M}_t = \{0\}, \quad t_1 > 0, \quad t_g = 1, \quad b_0 = v, \quad d_0 \geqq z_* - v \quad (\textit{trivially } d_0 \geqq w - v),$$

determine all its solutions satisfying $z(x) = z_*$.

But this is the zero-one linear programming problem with a specific way of obtaining the optimal solutions. It is sufficient to proceed only until in some k-th stage t_k-feasible solutions are found. If it is required to find only one optimal solution, we shall speak about Problem T1′.

Problem T11. *Solve Problem* T. *After an optimal solution has been found repeat the last, say k-th stage, with the modification*

$$d_0 = z_* + \delta|z_*| - v, \quad t_k = 1$$

and score all t_k*-feasible solutions. Here* $\delta > 0$ *is a given value.*

Hence, all suboptimal solutions to the zero-one linear programming problem over a given range of z are required.

Problem T2. *Let* $z_*^{(k)}$ *stand either for the minimal value of the objective function in Problem* $\mathrm{T}^{(k)}$ *or for* w *if* $\mathrm{T}^{(k)}$ *has no feasible solution. Let Problem* T *be subjected to the conditions*:

$$\mathcal{M}_t \subseteq \mathcal{M} - \{0\}, \quad \mathcal{M}_t \neq \emptyset, \quad b_0 \geqq z_*^{(1)} \quad (\text{trivially } b_0 \geqq w).$$

In the k-th stage $(k = 1, 2, \ldots)$ *determine all* t_k*-feasible solutions to minimize the objective function.* (*The bound* b_0 *may be adjusted to* $z_*^{(k-1)}$ *for* $k > 1$.)

This problem is a discrete analogy to the parametric right-hand side linear programming problem, which by itself determines its practical applicability.[1]) Let us point out that here a nonincreasing sequence $\{z_*^{(k)}\}$ may be systematically used to reduce the enumeration process, which is not possible if the way (2) mentioned at the beginning of this paragraph is used. When at most one of optimal t_k-feasible solutions is required we shall refer to the problem as T2′.

§ 2. ALGORITHM SBG

We are going to describe an algorithm for the solution of Problem T from § 1 based on the idea of Process SBG.[2]) The algorithm has a three-level cyclic structure. On the highest level the computation is divided into stages (index k). The second level is that of a basic cycle of the k-th stage, called φ-cycle (index r), which proceeds upon the elements of the sequence $\Phi^{(k-1)}$. Finally, at the third level we have ψ-cycles (index s) that modify the sequence $\Phi^{(k-1)}$. The notation relates to the fact that the modified form of $\Phi^{(k-1)}$ is read as Ψ until it obtains the definite name $\Phi^{(k)}$ at the end of the stage. Before we begin the description itself let us present two remarks about the formalism:

Operation $S \cup \{j\}$ where S stands for a sequence preserves the ordering as indicated — the element j is appended at the end of S.

The change of the iteration index (r or s) will always concern all sets and quantities even if some of them may remain unchanged at the beginning of the new iteration.

Preparation of computation. Set $\Phi^{(0)} = \{-(n+1)\}$.

The k-th stage. A sequence $\Phi^{(k-1)} = \{\varphi_u^{(k-1)}\}$ $(u \geqq 1,\ 1 \leqq k \leqq g)$ is given.[3])

φ-cycle.

Preparation part. Set $r = 0$, $l^0 = 0$, $\Gamma^0 = \emptyset$, $y_i^0 = b_i + t_g d_i$ $(i \in \mathscr{M})$.

Iteration part (r-th iteration). Given sequences $\Psi^r = \{\psi_1^r, \ldots, \psi_p^r\}$,[4]) Γ^r, and numbers l^r, y_i^r $(i \in \mathscr{M})$. Let us denote $\bar{y}_i^r = y_i^r - (t_g - t_k)\, d_i$ $(i \in \mathscr{M})$. Considering an element $\varphi_{r+1} \in \Phi^{(k-1)}$, distinguish the following four cases:

(1) $0 < \varphi_{r+1} \leqq n$. Increase the branching level:

$$\Gamma^{r+1} = \Gamma^r \cup \{\varphi_{r+1}\}\,, \quad \psi_{p+1}^{r+1} = \varphi_{r+1}\,,$$

$$l^{r+1} = l^r + 1\,; \quad y_i^{r+1} = y_i^r - a_{i\varphi_{r+1}}\ (i \in \mathscr{M})$$

---

[1]) Discretness of the parameter t means no restriction; rather the requirement $d_i \geqq 0$ does.

[2]) First published in a somewhat simpler form as $\varphi\psi$-algorithm in [3].

[3]) The stage number will not be quoted in indexing, as a rule.

[4]) p depends on r. Let $\Psi^0 = \emptyset$ by definition.

and continue in the φ-cycle by the next iteration.

(2) $\varphi_{r+1} > n$. Register the eliminated loop:

$$\Gamma^{r+1} = \Gamma^r \cup \{n - \varphi_{r+1}\}, \quad \psi_{p+1}^{r+1} = \varphi_{r+1},$$
$$l^{r+1} = l^r, \quad y_i^{r+1} = y_i^r (i \in \mathcal{M})$$

and continue in the φ-cycle by the next iteration.

(3) $-n \leqq \varphi_{r+1} \leqq 0$. Examine inequalities $\bar{y}_i^r \geqq 0$ $(i \in \mathcal{M})$.

(3a) If they hold, then score t_k-*feasible solution* and

($3a_1$) if $\varphi_{r+1} = 0$, continue in the φ-cycle by the next iteration;

($3a_2$) if $\varphi_{r+1} < 0$, enter the ψ-cycle.

(3b) If some of the inequalities does not hold, then put $\psi_{p+1}^{r+1} = \varphi_{r+1}$ and

($3b_1$) if $\varphi_{r+1} = 0$, continue in the φ-cycle by the next iteration;

($3b_2$) if $|\varphi_{r+1}| > l^r$, finish the k-th stage (the sequence $\Phi^{(k-1)}$ has been exhausted); if now $\Psi^{r+1} = \emptyset$, then the solution of the whole problem is finished; otherwise put $\Phi^{(k)} = \Psi^{r+1}$ and go on to the $(k+1)$-st stage;

($3b_3$) if $0 < |\varphi_{r+1}| \leqq l^r$, decrease the branching level, i.e. leave out the elements of Γ^r from the end through $|\varphi_{r+1}|$ positive ones, prefix the sign 'minus' to the last of them to obtain the sequence Γ^{r+1}, and put

$$l^{r+1} = l^r - |\varphi_{r+1}|, \quad y_i^{r+1} = y_i^r + \sum_{j_\varkappa > 0} a_{ij_\varkappa} \; (i \in \mathcal{M})$$

where the summation is considered over the positive elements $j_\varkappa$ left out from Γ^r; continue in the φ-cycle by the next iteration.

(4) $\varphi_{r+1} < -n$. Enter the ψ-cycle.

RETURN from the ψ-cycle has two alternatives: (a) Either the sequence $\Phi^{(k-1)}$ has been exhausted. Then finish the k-th stage; if $\Psi^{r+1} = \emptyset$, the problem is solved; else put $\Phi^{(k)} = \Psi^{r+1}$ and go on to the $(k+1)$-st stage. (b) Or the sequence $\Phi^{(k-1)}$ has not yet been exhausted. Then continue in the φ-cycle by the next iteration.

ψ-cycle.

Preparation part. Set $s = 0$, $\Psi^{r0} = \Psi^r$, $l^{r0} = l^r$, $\Gamma^{r0} = \Gamma^r$, $y_i^{r0} = y_i^r$ $(i \in \mathcal{M})$,

$$v_{l^r}^{r0} = \bar{v}_{l^r}^{r0} = |\varphi_{r+1}|_{(n)} \tag{2.1}$$

where the symbol on the right in (2.1) means $|\varphi_{r+1}|$ *modulo* n. If $-n \leqq \varphi_{r+1} < 0$, then go to CDBL (see below).

Iteration part (s-th iteration). Given sequences $\Psi^{rs} = \{\psi_1^{rs}, \ldots, \psi_q^{rs}\}$, [5]) Γ^{rs}, numbers $l^{rs} \equiv l$, y_i^{rs} $(i \in \mathcal{M})$, and sets $\{v_{l^r}^{rs}, \ldots, v_l^{rs}\}$, $\{\bar{v}_{l^r}^{rs}, \ldots, \bar{v}_l^{rs}\}$. Let us denote $\bar{y}_i^{rs} =$

[5]) q depends on r and s.

$= y_i^{rs} - (t_g - t_k)\, d_i$ $(i \in \mathscr{M})$. For the sake of simplicity, we shall omit the index r.

If $\bar{y}_i^s \geqq 0$ for all $i \in \mathscr{M}$ and if either $k = 1$, $s = 0$ or IBL preceded, then score a *t_k-feasible solution* $J_l^s = \{j \in \Gamma^s \mid j > 0\}$.

In any case, go on deriving from Γ^s the sets of elements $1 \leqq j \leqq n$

$$\Gamma_1^s = \{j \mid j \in \Gamma^s \quad \text{or} \quad -j \in \Gamma^s\}, \quad \Gamma_2^s = \{j \mid -(n+j) \in \Gamma^s\} \tag{2.2}$$

and forming two sets of "free" elements

$$N^s = \mathscr{N} - \Gamma_1^s, \quad \bar{N}^s = \mathscr{N} - (\Gamma_1^s \cup \Gamma_2^s). \tag{2.3}$$

If $N^s = \emptyset$, go to CDBL. If $\bar{N}^s = \emptyset$, then go to ODBL (see below). Otherwise *tests* are to be applied. We can use here any of the tests known from Balasian algorithms, but we apply them in two modes:

1-st mode – read the quantities y_i^s and the set N^s as input information. When some test succeeds CDBL follows. Otherwise two possibilities arise: if $\bar{y}_i^s = y_i^s$ $(i \in \mathscr{M})$, then go to IBL; if not, repeat the tests in

2-nd mode – read $\bar{y}_i^s$ and $\bar{N}^s$ on input. In case some test succeeds, ODBL follows; otherwise go to

IBL: Let

$$\begin{aligned} F^s &= \{j_1, \ldots, j_\varrho\}, \quad \bar{F}^s = F^s \cup \{j_{\varrho+1}, \ldots, j_{\bar{\varrho}}\}, \\ G^s &= \{k_1, \ldots, k_\sigma\}, \quad \bar{G}^s = G^s \cup \{k_{\sigma+1}, \ldots, k_{\bar{\sigma}}\} \end{aligned} \tag{2.4}$$

be sets which have the following meaning:

F^s contains elements which *must* be present in every combination $J \supset J_l^s$ so that it may be t_g-feasible $(F^s \subseteq N^s)$;

$\bar{F}^s$ analogously for t_k-feasibility to be guaranteed $(\bar{F}^s - F^s \subseteq \bar{N}^s)$;

G^s contains elements which *must not* be present in any combination $J \supset J_l^s$ so that it may be t_g-feasible $(G^s \subseteq N^s)$;

$\bar{G}^s$ analogously for t_k-feasibility to be guaranteed $(\bar{G}^s - G^s \subseteq \bar{N}^s)$. [6])

We suppose that the reduction possibilities which might follow from the extent of these sets (e.g. $\bar{N}^s - \bar{G}^s = \emptyset$) have been used in the tests.

(a) If $\bar{F}^s \neq \emptyset$, then put

$$\begin{aligned} \Gamma^{s+1} = \Gamma^s &\cup \{-k_1, \ldots, -k_\sigma\} \cup \{j_1, \ldots, j_\varrho\} \cup \\ &\cup \{-(n + k_{\sigma+1}), \ldots, -(n + k_{\bar{\sigma}})\} \cup \{j_{\varrho+1}, \ldots, j_{\bar{\varrho}}\}, \end{aligned} \tag{2.5}$$

$$\psi_{q+\varkappa}^{s+1} = n + k_\varkappa \quad (\varkappa = 1, \ldots, \sigma), \tag{2.6}$$

[6]) These sets can be obtained e.g. by means of Test BF [1].

(2.7) $$\psi^{s+1}_{q+\sigma+\varkappa} = j_\varkappa \quad (\varkappa = 1, \ldots, \bar{\varrho}),$$

(2.8) $$y_i^{s+1} = y_i^s - \sum_{j \in \bar{F}^s} a_{ij} \quad (i \in \mathcal{M}),$$

(2.9) $$\psi^{s+1}_{q+\sigma+\bar{\varrho}+1} = 0 \quad \text{if} \quad y_i^{s+1} \geqq 0 \text{ for all } i \in \mathcal{M}$$
$$\text{and } \bar{y}_i^{s+1} < 0 \text{ for at least one } i \in \mathcal{M},$$

(2.10) $$v^{s+1}_{l+\varrho} = v_l^s + \varrho,$$

(2.11) $$v^{s+1}_{l+\varrho+\varkappa} = 1 \quad (\varkappa = 1, \ldots, \bar{\varrho} - \varrho),$$

(2.12) $$\bar{v}^{s+1}_{l+\bar{\varrho}} = \bar{v}_l^s + \bar{\varrho},$$

(2.13) $$l^{s+1} = l^s + \bar{\varrho},$$

and continue in the ψ-cycle by the next iteration.

(b) If $\bar{F}^s = \emptyset$, then set $\varrho = 0$, $\bar{\varrho} = 1$, define the element j_1 by the relation

(2.14) $$v^s_{j_1} = \max_{j \in N^s - G^s} v_j^s$$

where

$$v_j^s = \sum_{i \in \mathcal{M}} \min \{\bar{y}_i^s - a_{ij}, 0\},$$

apply formulas (2.5)–(2.9), put

(2.15) $$v^{s+1}_{l+1} = \bar{v}^{s+1}_{l+1} = 1, \quad l^{s+1} = l^s + 1,$$

and continue in the ψ-cycle by the next iteration.

CDBL: Leave out the elements $\psi^s_q, \psi^s_{q-1}, \ldots$ until either

(a) an element of the type $\psi \leqq 0$, say $\psi^s_{q-\gamma} (\gamma \geqq 0)$, is encountered. Then substitute

(2.16) $$\psi^{s+1}_{q-\gamma} = \psi^s_{q-\gamma} - (v_l^s - \mu^s)$$

where μ^s is the number of elements $0 < \psi \leqq n$ left out before $\psi^s_{q-\gamma}$ has appeared $(0 \leqq \mu^s < v_l^s)$ and

(a_1) if $l < v_l^s$, put $\Psi^{r+1} = \Psi^{s+1}$ and RETURN to the φ-cycle (the sequence $\Phi^{(k-1)}$ is exhausted);

(a_2) if $l \geqq v_l^s$, leave out the elements of Γ^s from the end through v_l^s positive ones, prefix the sign 'minus' to the last of them to get the sequence Γ^{s+1}, and put

(2.17) $$l^{s+1} = l^s - v_l^s, \quad y_i^{s+1} = y_i^s + \sum_{j_\varkappa > 0} a_{ij_\varkappa} \quad (i \in \mathcal{M})$$

where the summation is considered over the positive elements $j_\varkappa$ left out from Γ^s. If $l^{s+1} < l^r$, then set $\Psi^{r+1} = \Psi^{s+1}$, $l^{r+1} = l^{s+1}$, $y_i^{r+1} = y_i^{s+1} (i \in \mathcal{M})$ and RETURN to the φ-cycle; else continue in the ψ-cycle by the next iteration; or

(b) no element of the type $\psi \leqq 0$ is encountered, but

(b_1) the sequence Ψ^s is exhausted.[7]) In this case set $\Psi^{r+1} = \emptyset$ and RETURN to the φ-cycle (the sequence $\Phi^{(k-1)}$ has been exhausted);

(b_2) v_l^s elements of the type $0 < \psi \leqq n$ are left out; let the last of them be $\psi_{q-\delta}^s$. Substitute

$$\psi_{q-\delta}^{s+1} = n + \psi_{q-\delta}^s \tag{2.18}$$

and go on as in the case (a_2) above.

ODBL: Put

$$\psi_{q+\varkappa}^{s+1} = -(n + v_{l-\varkappa+1}^s) \quad \text{for} \quad \varkappa = 1, 2, \ldots \quad \text{until} \quad \sum_{\varkappa \geqq 1} v_{l-\varkappa+1}^s = \bar{v}_l^s. \tag{2.19}$$

If $l < \bar{v}_l^s$, then set $\Psi^{r+1} = \Psi^{s+1}$ and RETURN to the φ-cycle (the sequence $\Phi^{(k-1)}$ has been exhausted). Otherwise proceed as in the case (a_2) in CDBL replacing v_l^s by $\bar{v}_l^s$.

The description of Algorithm SBG is now complete; its adaptations for the solution of the special problems follow.

An adaptation for Problem T1 – *Algorithm* SBG1. If in the k-th stage a t_k-feasible solution has been obtained (either in (3) of the φ-cycle or at the beginning of the ψ-cycle), substitute new values $\tilde{y}_0^r = 0$ or $\tilde{y}_0^s = 0$, $\tilde{t}_k = 1 = t_g$ and continue the iteration. The variable y_0^r or y_0^s is then to be modified in the same way whenever a feasible solution is reached. Also replace the set N^s by $\bar{N}^s$ where it concerns CDBL, i.e. in examining $N^s = \emptyset$ after (2.3) and in the 1-st mode of tests.

Concerning Problem T1′ the only difference is in the substitution which now reads $\tilde{y}_0^r = -\varepsilon$ or $\tilde{y}_0^s = -\varepsilon$ where ε is a given positive number not exceeding the least possible variation of the value of the objective function.

An adaptation for Problem T11 – *Algorithm* SBG11. Proceed as in Algorithm SBG1 until an optimal solution is reached. Then repeat the last, say the k-th stage,[8]) with $y_0^0 = z_*^{(k)} + \delta|z_*^{(k)}|$, $t_k = 1$ (now the quantity y_0 will no longer be changed after obtaining a feasible solution).

An adaptation for Problem T2 – *Algorithm* SBG2. The section (3b) of the φ-cycle is to be modified as follows: If $\bar{y}_0^r < 0$,[9]) then proceed as in (3a); else let it in the original reading. In the k-th stage $(k > 1)$ set $y_0^0 = z_*^{(k-1)}$ in Preparation part of the φ-cycle. Substitute a new value $\tilde{y}_0^r = 0$ or $\tilde{y}_0^s = 0$ always after a t_k-feasible solution is obtained. For Problem T2′ again with the change: $\tilde{y}_0^r = -\varepsilon$, $\tilde{y}_0^s = -\varepsilon$.

The purpose of the modification of (3b) is this: It could happen that some t_k-feasible solutions, having been registered in the sequence Φ on early stages, yield $\bar{y}_0^r < 0$ when their "turn" comes in the k-th stage. If it were not for this modification, they

[7]) This covers also the trivial case $\Psi^s = \emptyset$ (at the beginning of the process).

[8]) On the basis of $\Phi^{(k-1)}$, of course.

[9]) Be aware of $\bar{y}_0^r = y_0^r$ because of $d_0 = 0$.

would remain in the sequence till the end of the computation preventing the sequence from being shortened by the mechanism CDBL.

Summarizing our knowledge about the end of the algorithms, we can distinguish:

(a) *normal end* when the last stage runs throughout with $\bar{y}_i^s = y_i^s$ $(i \in \mathcal{M})$. It occurs – if the singular case $d_i = 0$ $(i \in \mathcal{M})$ is neglected – for $k = g$ in one-staged Problem T1, multi-staged T1 with no feasible solutions, T11 (the repeated stage), or T2. There is always $\Psi^{r+1} = \emptyset$.

(b) *abnormal end* when the last stage begins with $\bar{y}_i^s < y_i^s$ for some $i \in \mathcal{M}$. This is the case of Problems T1 and T11. The resulting sequence Ψ^{r+1} then may contain a reminder from the beginning phase of the last stage.

Finally, we shall summarize the measures which prevent Algorithm SBG from multiple occurence of t_k-feasible solutions:

In the φ-cycle, the registration of a t_k-feasible solution is deleted when unnecessary: in the case $(3a_1)$ the element $\varphi_{r+1} = 0$ does not enter the new sequence $\Phi^{(k)}$, in the case $(3a_2)$ $\varphi_{r+1} = 0$ is eliminated by means of the mechanism CDBL.

In the ψ-cycle, (1) the nonredundant nature of the enumeration process shows, (2) t_k-feasible solutions are accepted after IBL only, (3) solely t_{g-k}-feasible solutions are registered in $\Phi^{(k)}$ (see formula (2.9)). Measure (2) is meant word for word, thus excluding scoring a t_k-feasible solution also in the 0-th iteration of the ψ-cycle (except for the 1-st stage). Thereby it is impossible that the scoring could be repeated at the beginning of the ψ-cycle even in such rather unlikely situations: (a) Just before entering the ψ-cycle a t_k-feasible solution is indicated in the section (3) of the φ-cycle. (b) A t_h-feasible solution was scored in the h-th stage $(h < k)$ immediately before creating a gap (ODBL) in the ψ-cycle. When this configuration, later registered in the sequences Φ, remains unchanged up to the k-th stage and (in the case of SBG2) y_0 is not nullified, then the very same combination meets successively as $t_{h+1}, \ldots, t_k$-feasible at the beginning of the ψ-cycle.

§ 3. JUSTIFICATION OF THE ALGORITHM

We shall concede Algorithm SBG to be justified enough if we express it in terms of Process B and Algorithm BG. In the k-th stage Algorithm SBG produces a sequence $\Phi^{(k)}$ on the basis of $\Phi^{(k-1)}$. We are going to prove[10]) about it a

Lemma. *The sequence* $\Phi^{(k)}(1 \leqq k \leqq g)$ *registers Process* B *for Problem* $T^{(k)}$ *so that*

1. *elements* $0 < \varphi_{r+1} \leqq n$ *represent* IBL – *see the case* (1) *in the* φ*-cycle*;
2. *elements* $\varphi_{r+1} > n$ *indicate eliminated closed branches* – *see the case* (2);
3. *elements* $\varphi_{r+1} < 0$ *represent* DBL; *among them*

[10]) The proof is also intended to serve as an explanation to the algorithm.

a) *elements* $-n \leqq \varphi_{r+1} < 0$ *indicate* CDBL *following after a recorded* t_{g-k}*-feasible solution — see the case* (3). *The quantity* $|\varphi_{r+1}|$ *states how much the branching level is to be decreased*;

b) *elements* $\varphi_{r+1} < -n$ *indicate* ODBL — *see the case* (4). *The quantity* $|\varphi_{r+1} + n|$ *states how much the branching level is to be decreased*;

4. *elements* $\varphi_{r+1} = 0$ *record* t_{g-k}*-feasible solutions — see the case* (3);

5. *all removable closed branches are eliminated.*

Proof. We shall proceed using mathematical induction. Let us assume that the assertion of Lemma holds for the sequence $\Phi^{(k-1)}$ $(k > 1)$. Process B for Problem $T^{(k)}$ can be derived from the process for $T^{(k-1)}$ by its extending at the points where it was interrupted because of the conditions on $T^{(k-1)}$ (ODBL). The mutual relation of the problems makes it possible.

In the φ-cycle we realize Process B on the basis of the sequence $\Phi^{(k-1)}$ examining its elements φ_{r+1} for $r = 0, 1, \ldots$. At the r-th iteration the following sets and quantities are available: a sequence Γ^r, a combination J^r consisting of the positive elements of Γ^r, a branching level l^r (i.e. the number of elements taken in the combination J^r), and 'slack' variables

$$y_i^r = b_i + t_g d_i - \sum_{j \in J^r} a_{ij}\,, \quad \bar{y}_i^r = b_i + t_k d_i - \sum_{j \in J^r} a_{ij} \quad (i \in \mathcal{M})\,.$$

These all are adjusted in (1), (2), and ($3b_3$) in the way known from Algorithm BG. At the same time a new sequence $\Phi^{(k)}$ is constructed; its intermediate state is referred to, for formal reasons, by Ψ^r.

In the cases (1), (2), and (3b) the element φ_{r+1} is kept for the future stage, i.e. $\Psi^{r+1} = \Psi^r \cup \{\varphi_{r+1}\}$.

In the case ($3b_2$) the inequality $|\varphi_{r+1}| > l^r$ signals the end of Process B for Problem $T^{(k-1)}$ at this point — thus φ_{r+1} must be the last element of $\Phi^{(k-1)}$.

In the case ($3a_1$) when a t_k-feasible solution has been identified there is no need to keep it further, therefore φ_{r+1} is suppressed, i.e. $\Psi^{r+1} = \Psi^r$.

The cases (4) and ($3a_2$) lead to the ψ-cycle — the part of the algorithm where the sequence Ψ^r is substantially modified: either extended by "drawing out" Process B or reduced by eliminating removable closed loops. Here are some more details:

We shall use the index $s = 0, 1, \ldots$ to number the iterations of the "new" part of Process B upon a gap (the index r will not be written provided that no confusion arises). After s iterations have been executed the state of Process B is determined by the set Γ^s (J^s) and by the quantities l^s, y_i^s, $\bar{y}_i^s$ $(i \in \mathcal{M})$. We shall present only a few explanating remarks concerning those parts of the ψ-cycle which are obviously derived from Algorithm BG.

The sequence Γ^s is developed mainly in the section IBL. Formula (2.5) appends to it elements of three types: j from the set $\bar{F}^s$, $-j$ for $j \in G^s$, and $-(n + j)$ for $j \in$

$\in \bar{G}^s - G^s$. Decreasing of branching level causes a reduction of the sequence from the end; the last element left turns to the type $-j$.

Rather peculiar role of the elements $-(n+j)$ arises from the fact that they cannot be present in any t_k-feasible combination, but must be taken into account for t_{q-k}-feasible combinations. This stipulates the role of the set Γ_2^s for the distinction between N^s and $\bar{N}^s$ – see (2.2), (2.3).

The quantities v_l^s, $\bar{v}_l^s$ control the manipulation with the groups of elements of Γ^s during DBL. The number v_l^s states by what the given level l is to be decreased in the case of CDBL; it is cumulated according to (2.10) if $F^s \neq \emptyset$; "empty" value is 1. The number $\bar{v}_l^s$ has an analogous function with regard to $\bar{F}^s$ and ODBL. If $\bar{F}^s = \emptyset$, the element for branching is chosen according to the criterion (2.14) due to Balas.

The registration of Process B in the ψ-cycle continues as a sequence Ψ, now with indexing Ψ^{rs} (briefly Ψ^s). Let us assume that the sequence Ψ^s has the properties declared by Lemma for $\Phi^{(k)}$. We shall prove that the same is true for the sequence Ψ^{s+1}. Let us concentrate on the branching mechanisms, which are those parts of the ψ-cycle that can influence our sequence:

IBL: Here merely an extension of Ψ^s may take place, namely in formula (2.6) where an obvious fact is utilized that elements of G^s function identically as indications of removed closed loops, further in formula (2.7) where elements for branching are recorded, and in formula (2.9) where a t_{y-k}-feasibility is registered. All is in accordance with the way of registration according to Lemma.

CDBL: Here the sequence Ψ^s is modified (reduced, as a rule). The branching level can be decreased by v_l^s and the ending part of Ψ^s which represents a removable closed loop can be eliminated. Therefore positive elements starting from the end of Ψ^s are omitted. Branching levels are counted on elements of the type $0 < \psi \leqq n$. The algorithm treats differently two alternatives:

(a) Dropping out the positive elements is stopped when an element of the type $\psi \leqq 0$ occurs. According to the assumption about Ψ^s the quantity $|\psi_{q-\gamma}^s|_{(n)}$ prescribes the number of levels to be decreased (in the closed way when $-n \leqq \psi \leqq 0$, and open way when $\psi < -n$). After an adjustment according to equation (2.16) it will have the same meaning. Really, since only $\mu^s < v_l^s$ levels were accompanied with elimination when decreased, there will be no other branching on the level $l^s - \mu^s$ in the next stage; therefore the eliminating of a loop need not be indicated and the remainder $v_l^s - \mu^s$, "unused" for elimination, can be added to the element $\psi_{q-\gamma}^s$.

Notice that after we apply formula (2.16) the element $\psi_{q-\gamma}^s = 0$ becomes an element of the type $-n \leqq \psi < 0$. A loop containing such element may be eliminated only after the combination marked by it has been recognized as t_k-feasible in $(3a_2)$ of the φ-cycle. CDBL is initiated for that in Preparation part of the ψ-cycle.

The nature of Process B implies that if it were $v_{l^r}^{r0} = 0$, then for some $s' \geqq 0$ it would be $l^{s'+1} = l^r$ and DBL would take place in the s'-th iteration of the ψ-cycle. Were it CDBL, one would re-enter the sequence $\Phi^{(k-1)}$, decrease the branching level by $|\varphi_{r+1}|_{(n)}$, and eliminate. But to this aim, as our algorithm actually does it, one can

make use of the elimination mechanism of the ψ-cycle as well as the formal identity of both the new and the old parts of the sequence Φ, here unified under the notation Ψ. Moreover, (2.1) makes it possible to link together the old and the new parts so that the quantity v^{r1} can be accumulated on v^{r0} (then $l^{s'} > l^r$, $l^{s'+1} < l^r$) and the remainder $v^{s'} - \mu^{s'}$ from the last decrease can be added to $\psi^{s'}_{q-\gamma}$ (the element φ_{r+1} is no longer considered). Now, after RETURN to the φ-cycle the r-indexing again continues: $\Psi^{r+1} = \Psi^{r,s'+1}$. The next element of $\Phi^{(k-1)}$ to be examined is φ_{r+2}.

Parallelly to the modification of Ψ^s, the sequence Γ^s along with the appropriate quantities are modified according to the rule of decreasing a branching level by v^s_l in Algorithm BG – see formula (2.17). Exhausting Γ^s indicates that Process B for Problem $T^{(k)}$ came to an end – see the case (a_1). Then φ_{r+1} is the last element of $\Phi^{(k-1)}$. Indeed, the inequality $v^s_l > l$ implies[11]) $|\varphi_{r+1}|_{(n)} > l^r$ so that φ_{r+1} must have completed the preceding stage.

(b) The elimination of a loop proceeds as far as possible – through all v^s_l levels. The element $\psi^{s+1}_{q-\delta}$ formed according to formula (2.18) indicates the removed loop in accordance with the assertion of Lemma. The extreme situations (b_1) and $l^{s+1} < l^r$ can be treated similarly to the case (a): either as the exhausting of Γ^s – compare (a_1), or RETURN to φ-cycle – compare (a_2).

Since Ψ^s does not contain any removable closed loops and each newly obtained such loop is, as shown above, immediately eliminated in the greatest possible measure, neither Ψ^{s+1} will contain such loops.

ODBL: Here the sequence Ψ^s is extended. The branching level can be decreased by $\bar{v}^s_l$, but it is necessary that all the intermediate levels on which in succeeding stages another branching can take place be recorded. Clearly, formula (2.19) meets this demand. The extreme situations $l < \bar{v}^s_l$ and $l^{s+1} < l^r$ are again analogous to those in section CDBL.

Now we can proceed to the conclusion. Since $\Psi^{r0} = \Psi^r$ and the assertion of Lemma about Ψ^r holds when the ψ-cycle appears for the first time (recall the analysis of the φ-cycle), the assertion is thereby proved for the whole $\Phi^{(k)}$ as well. And, concerning the completion of our induction: The sequence $\Phi^{(1)}$ is totally built in the ψ-cycle ($\Phi^{(0)}$ is merely a formal tool of initiating the staged process[12])). To prove the validity of Lemma for it, we may use the previous analysis of the ψ-cycle observing that in the 0-th iteration one of the following possibilities can take place:

IBL – the assertion certainly holds;
CDBL – the process finishes via (b_1);[13])
ODBL – there is $\psi^1_1 = -(n+1)$, i.e. $\Phi^{(1)} = \Phi^{(0)}$.

[11]) Compare (2.1), (2.10), (2.11), (2.13), and (2.17).

[12]) However, the choice $\varphi^{(0)}_1 = -(n+1)$ is not arbitrary; it guarantees the setting $v^0_0 = \bar{v}^0_0 = 1$ by means of (2.1).

[13]) See footnote 7 in § 2.

The proof of Lemma is now completed. Let us append a few words about the termination of the algorithm. This happens after finishing that k-th stage in which all registered t_g-feasible solutions were identified as t_k-feasible and no ODBL took place, i.e. the resulting sequence Ψ^{r+1} does not contain any elements $\psi \leqq 0$. But then all loops of this sequence have been removed (see Lemma). The only possible termination of this kind is that in the section CDBL, case (b_1) – hence $\Psi^{r+1} = \emptyset$.

Remark. In a perfectly stratified process SBG no elements of type $-n \leqq \varphi \leqq 0$ are generated.

An explanation to the adaptations of Algorithm SBG remains to be added. In all of them a conception introduced in [1] is used: The objective function is considered a constraint – including the use of the term 'feasibility' – however with a variable right-hand side which is represented by the current minimal value of the objective function.

Let us have the s-th iteration of the ψ-cycle in Algorithm SBG1 and denote $z^s = \sum_{j \in J^s} a_{0j}$ so that $y_0^s = b_0 + d_0 - z^s$. [14]) If J^s is t_k-feasible, the adjustment $\tilde{y}_0^s = 0$ can be interpreted as a definition of a new $\tilde{d}_0 = z^s - b_0$ and $\tilde{t}_k = 1$ as $\tilde{g} = k$. Then in the succeeding iterations a combination J^u will be recorded as t_k-feasible only if it satisfies

$$\tilde{y}_0^u = y_0^u \geqq 0\,, \quad \text{i.e.} \quad z^u \leqq b_0 + \tilde{d}_0 = z^s\,;$$

but this is enough with respect to the purpose of Problem T1. This justifies, at the same time, the adjustment of the algorithm concerning the sets N^s and $\bar{N}^s$. (Such adjustment is useful because of the possibility of $\bar{N}^s \neq N^s$ in this case.)

To Algorithm SBG2: Setting $y_0^0 = z_*^{(k-1)}$ may be regarded as a redefinition $b_0 = z_*^{(k-1)}$. In this way all combinations giving $y_0^s = z_*^{(k-1)} - z^s < 0$ are suppressed as infeasible ones, beginning from the k-th stage. The modification of section (3b) in the φ-cycle makes it possible to remove these infeasible solutions even from the sequence Φ as soon as possible. The measure after obtaining a t_k-feasible solution can be considered a restatement $\tilde{b}_0 = z^s$. In succeeding iterations then $y_0^u \geqq 0$ implies $z^u \leqq \tilde{b}_0 = z^s$. [14])

§ 4. REALIZATION OF THE ALGORITHM

Algorithm SBG together with the adaptations was programmed experimentally on IBM/360 (in PL/I). Here we present some remarks regarding this particular realization.

1. Test BF [1] is used both for testing and for preparing the sets (2.4). The second mode of the test starts from the terminal state of the first one. Both modes are inter-

[14]) Similarly for the r-th iteration of the φ-cycle.

preted for a great part by one sequence of instructions (see [2]).

2. We mentioned in § 1 a good stratifying of the staged enumeration process as a condition of its efficiency. Test BF makes it possible that even the *perfect stratifying* can be achieved by means of only a formal adjustment of a problem — by the transformation $x'_j = 1 - x_j$. It can always be done in Problem T1 or T11 because it suffices to guarantee $a'_{0j} \geqq 0$ $(j \in \mathcal{N})$. Similarly in Problem T2 provided that the coefficients a_{ij} $(i \in \mathcal{M}_t)$ have the same sign for each $j \in \mathcal{N}$.

3. The sequences Φ and Ψ cannot be handled without using external storage devices. Therefore we assign them *files* FI and PSI. File FI is processed sequentially, but on PSI reading and writing alternate all the time (IBL, CDBL, ODBL), which suggests operations to be organized in the following way: Let PSIW identify the working area (buffer) reserved for items of PSI in the core storage. Each time PSIW has been filled up or empted we transfer the front of file PSI to the middle of the working area. Thereby we reduce the danger that oscillations of the front of PSI will cause too frequent exchange of information between internal and external devices. The sequence $\Phi^{(k-1)}$ is replaced by $\Psi^{r+1} \equiv \Phi^{(k)}$ simply by overnaming the appropriate files.

Continual checking is necessary to avoid the possibility that a reserved storage capacity is not sufficient for the storing of file PSI. Let p be a current number of elements of a sequence Ψ and N the capacity available. If $p = N - n$, then for Problem T1 (T11): Set $\tilde{t}_k = 1$ and continue in iterations. Thereafter only IBL and CDBL may occur so that the number of elements of PSI will never exceed $p + n = N$. For Problem T2: terminate. Thus, lack of storage does not lead, at any case, to the loss of the computation, but only in case T1 staging is suppressed, and in case T2 an intermediate result is obtained (which might be completed later by a new computation).

4. The classical method of linear programming (with the "continuous" condition $0 \leqq x_j \leqq 1$) can serve as a useful tool for the formulation of our problems. Particularly, for Problem T1 it gives a necessary condition of its solvability and if $\tilde{z}_*$ is the optimal objective function value for the "continuous" problem, then t_1 should be determined to satisfy

$$b_0 + t_1 d_0 \geqq \tilde{z}_*$$

and a constraint

$$-\sum_{j=1}^{n} a_{0j} x_j \leqq -\tilde{z}_*$$

may be appended. For Problem T2 the parametric technique could help to prepare t_1 and t_g with respect to the solvability of the problem.

Some numerical experiments with Algorithms SBG are quoted in [2]. So far there have been few of them to allow more general conclusions. In addition, it appears necessary that for practical use the algorithms should be equipped with more means

of reduction. The paper [1, § 3] suggests some possibilities in this respect, others arise from Glover's and Geoffrion's idea of surrogate constraints (see [1] for references).

References

[1] *Hrouda, J.:* A contribution to Balas' algorithm. This issue, 336—353.
[2] *Hrouda, J.:* Tři příspěvky k bivalentnímu lineárnímu programování. Příloha k výzkumné zprávě VZ-321/70. VÚTECHP, Praha 1970.
[3] Výzkumná zpráva VZ-124/68 (řešitel J. Hrouda). VÚTECHP, Praha 1968.

Souhrn

ETAPIZACE V BALASOVĚ ALGORITMU

JAROSLAV HROUDA

V článku je rozvinuta myšlenka: rozčlenit enumerační proces na části, nikoliv ve smyslu horizontálním — dekomposice úlohy, nýbrž ve smyslu vertikálním — rozvrstvení enumeračního procesu. Takové členění (etapizace) vzniká: (1) při zkusmém omezování minimální hodnoty účelové funkce v úloze bivalentního lineárního programování, (1a) jsou-li žádána všechna suboptimální řešení v daném pásmu hodnot účelové funkce, (2) při zavedení diskrétního parametru do pravých stran.

Náš způsob řešení takto členěných úloh lze charakterizovat jako rozšířené použití Gloverovy evidence enumeračního procesu B [1]. Ta se nyní zaznamenává do „dlouhé" posloupnosti, v níž jsou navíc registrovány údaje o přerušeních procesu z důvodu etapizace. Posloupnost se samočinně zbavuje úseků, které pro pozdější etapy nemají význam. Na tuto ideu můžeme tedy pohlížet jako na záznamovou metodiku — etapový enumerační proces BG — dovolující obecnější použití než prostý enumerační proces (především možnost parametrizace v úloze bivalentního lineárního programování).

V § 1 je vysvětlen princip etapového procesu BG (zkráceně: proces SBG); uvádí se jeho motivace, buduje terminologie a formulují speciální úlohy (viz (1), (1a) a (2) výše). V § 2 je detailně popsán algoritmus zvaný SBG, který uskutečňuje proces SBG, a jeho úpravy pro jednotlivé speciální úlohy. Ideově i formálně je poplatný algoritmu BG z [1]. V § 3 je algoritmus SBG odůvodněn tak, že je převeden k matematicky jednoduššímu algoritmu BG. V § 4 jsou dána některá doporučení stran realizace algoritmu.

Author's address: Jaroslav Hrouda, prom. mat., Výzkumný ústav technicko-ekonomický chemického průmyslu, Štěpánská 15, Praha 2.