INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS,
COMPUTATIONAL SCIENCE AND SYSTEMS ENGINEERING
DOI: 10.37394/232026.2023.5.20

Optimal operating mode for modular enumeration technology

VITALY O. GROPPEN
Data Processing Dept.,
North-Caucasian Institute of Mining and Metallurgy
(State Technological University),
Nikolajev str. 44, Vladikavkaz, 362021,
RUSSIA

Abstract: – The technology of modular enumeration is based on the minimizing of repetitive calculations - their results are stored in RAM and used as needed. While this technology can be used for different types of problems to be solved, optimal parameters of modular enumeration operating mode are known only in the case of solving discrete optimization problems with Boolean variables. The paper contains proofs of two theorems, permitting modular enumeration optimal operating mode determination in general case. The paper also contains examples of solution by modular enumeration optimal organization related to the two problems i. e. the search for the numerical value of multiple integral and that for a globally optimal solution to an extreme problem with Boolean variables, the last being based on knapsack problem solution.

Key-Words: - modular enumeration, operating mode, optimal parameters, examples, multiple integrals, knapsack problem, theorems.

Received: October 16, 2022. Revised: September 15, 2023. Accepted: October 19, 2023. Published: November 24, 2023.

1 Introduction

All the numerical problems we deal with in the first approximation can be divided into two groups. The first group consists of problems solved by efficient algorithms with a polynomial dependence of the operating time on the size of the task data [1, 2], while for problems of the second group such algorithms have not been developed, and their solution time is known to be related to their data size according to the exponential law [3-5]. As examples of the second group of problems, we can also suggest the search for numerical values of multiple integrals, roots of equations, discrete programming problems, where the search for solutions in the general case is based on various enumeration procedures [6, 7, 10-16]. These leads to the efforts to reduce the running time when solving numerical problems of the second group which can be identified as either analytical or computer-based. The latter dominate either in relation to the use of parallel computing [8, 9], or in relation to special procedures for a single processor, permitting to reduce the enumeration time. Thus, in the 20-th century for solving discrete programming problems, implicit enumeration methods were developed based on minimizing of enumeration volume, such as dynamic programming, branch-and-bound (B&B) methods, backtracking [9-11] and their modifications [12] – [15]. The technology of modular enumeration [16-19] first proposed in 2021 also for solving discrete programming problems is based on a different ideology i. e. calculation time shortening is achieved not by reducing the amount of enumeration, but by minimizing repetitive calculations - their results are stored in RAM and used as needed.

Both approaches though, have their drawbacks:

- Using the implicit enumeration, it is impossible to predict *a priori* either the number of iterations, or its upper bound and, as a result, it is impossible to predict the gain in time as a result of these procedures using as compared to the brute force method.
- Gain in running time by using modular enumeration as compared to the brute force method depends on its parameters, but optimal parameters of modular enumeration operating mode are known only in the case of solving discrete optimization problems with Boolean variables [17-20].

At the same time, unlike the implicit enumeration methods, modular enumeration permits us:

a) to use this technology for the solution of different kinds of problems, such as numerical calculation of integrals and roots of equations [6];
b) to predict the gain in time as result of its use as compared to the brute force method.

The paper aims at determination of modular enumeration optimal operating mode in general case, it contains description, mathematical modeling, analysis, and examples of modular enumeration optimal organization related to the two problems to be solved that is a search for the numerical values of multiple integrals [6] and search for a globally optimal solution to extreme problems with Boolean variables, the last being based on knapsack problem solving [21].

2 Designations and Assumptions

Q – enumeration volume;

m – the number of modules used (below we suppose that the value $\sqrt[m]{Q}$ is always integer);

q_i – size of the content of i-th module ($1 \le i \le m$);

V – used by modular enumeration software RAM size;

$\mathcal{V}$ - computer free RAM size;

ε – constant in the range $1 \div \sqrt[m]{Q}$;

k_j – j-th coefficient.

Below are used the following assumptions:

$$Q=\prod_{i=1}^{i=m} q_i. \quad (1)$$

$$V=k_1\sum_{i=1}^{i=m} q_i. \quad (2)$$

3 Basic Ideas of Modular Enumeration

The essence of solving any problem by modular enumeration technology is to implement its two stages: the first stage includes the preparation for the enumeration minimization of its repetitive computations, at the second stage the actual enumeration is carried out. Each stage consists of two steps. The first, preparatory stage consists of the following two steps:

At the first step all different components of enumeration units are grouped and stored into "m" modules thus creating q_i values ($1 \le i \le m$) satisfying (1). At the second step of the first stage for each unite of each module, its' part of the corresponding to the solved problem functions are calculated and stored as satisfying (2).

The second stage also includes two steps:

The first step relates to the usage of contents of the modules to generate all the Q values which correspond to the solved problem functions.

The second step includes processing of the data created at the previous step according to the specific of the problem solved. For example, their comparison in the case of extremum search and calculation of their sum in the case of integral value determination.

It should be noted that if at the second stage of modular enumeration an extremum is sought, then at the first stage after the second step it is possible to cut off the obviously "unpromising" components of each module, which will reduce the duration of the second stage, but at the same time increase the duration of the first one. An example is given in Section 4.

Example 1.

By the use of fixed Q value and m=2 modules of modular enumeration the following problem:

$$S = \iint_{a\ c}^{b\ d} [f_1(x_1)+f_2(x_2)]dx_1dx_2, \quad (3)$$

is substituted by the expression $S_1 \approx S$:

$$S_1=G \cdot \sum_{i=0}^{\sqrt{Q}} \sum_{j=0}^{\sqrt{Q}} [F_1(i)+F_2(j)], \quad (4)$$

where $G=\frac{(b-a)\cdot(d-c)}{[\sqrt{Q}+1]^2}$; $F_1(i)= f_1\{ a+(b-a)\cdot i/\sqrt{Q}\}$; $F_2(j)= f_2\{c+(d-c)\cdot j/\sqrt{Q}\}$.

During the first step of the first stage are created two modules, each i-th module containing $1+\sqrt{Q}$ different values of x_i (i=1,2) in the range presented in (3). During the second step values $x_{g,h}$, (g=1,2; $0 \le h \le \sqrt{Q}$) are substituted by the $1+\sqrt{Q}$ values of $f_g(x_{g,h})$. The latter are used at the first step of the second stage for sum W calculation:

$$W=\sum_{i=0}^{\sqrt{Q}} \sum_{j=0}^{\sqrt{Q}} \sum_{g=1}^{g=2} [f_g(x_{g,i})+f_{3-g}(x_{3-g,j})]. \quad (5)$$

The last step of the second stage results in value S_1 determination: $S_1=\frac{(b-a)\cdot(d-c)}{[\sqrt{Q}+1]^2} \cdot W.$ (6)

It is easy to show that the upper bound of the gain η in the running time for calculating (4) when applying modular enumeration with m=2 and coinciding sizes

of modules if compared to the traditional enumeration scheme is equal to the ratio:

$$\eta=\frac{[\sqrt{Q}+1]^2}{2[\sqrt{Q}+1]}=0.5\cdot[\sqrt{Q}+1]. \quad (7)$$

It is shown below that (7) corresponds to the optimal operating mode of modular enumeration in relation to the conditions for problem (3) solving if true is the following inequality: $\mathcal{V} \geq k_1 \cdot 2 \cdot \sqrt{Q}$. Experimental verification of this approach effectiveness for the case, when $f_1(x_1)=x_1^2$, $f_2(x_2)=\sqrt{x_2}$, *a=c=0, b=d=1,* m=2, is presented in [19].

4 Optimal Operating Mode Search

The description of modular enumeration presented above poses two questions about its optimal operating mode:

- What is the optimal number of modules, minimizing the running time for a specific task?
- What should be the optimal q_i (i=1,2,…,m) values for fixed number of modules m and Q value, minimizing needed for any problem solving free RAM size?

To answer the second question its formal statement using (1) and (2), where m and Q are constants is analyzed below:

$$\sum_{i=1}^{i=m} q_i \rightarrow \min; \begin{cases} \prod_{i=1}^{i=m} q_i = Q; \\ \forall i,\ q_i \geq 0. \end{cases} \quad (8)$$

True is the following theorem, solving (8):

Theorem 1. Optimal solution of (8) should satisfy the following conditions: $\forall i,\ q_i=\sqrt[m]{Q}$. (9)

Proof. Obviously, if the theorem is true, then modules of which satisfies condition (9):

$$V_1 = k_1 \cdot m \cdot \sqrt[m]{Q}. \quad (10)$$

Let now determine a new amount of required RAM V_2 with the following modules sizes:

$$\begin{cases} q_i=\sqrt[m]{Q}-\varepsilon; \\ q_j=\left(\sqrt[m]{Q}\right)^2/\left(\sqrt[m]{Q}-\varepsilon\right); \\ \forall t\neq(i \vee j):\ q_t=\sqrt[m]{Q}. \end{cases} \quad (11)$$

Obviously, if the theorem 1 is true, then $V_2 \geq V_1$. (12)

The value V_2 corresponding to (11) is equal to:

$$V_2=k_1\cdot\left[\sqrt[m]{Q}-\varepsilon+\left(\sqrt[m]{Q}\right)^2/\left(\sqrt[m]{Q}-\varepsilon\right)+(m-2)\cdot\sqrt[m]{Q}\right]. \quad (13)$$

We denote the difference V_2-V_1 as ΔV: $\Delta V= k_1\cdot\left[\sqrt[m]{Q}-\varepsilon+\left(\sqrt[m]{Q}\right)^2/\left(\sqrt[m]{Q}-\varepsilon\right)-2\cdot\sqrt[m]{Q}\right]$. (14)

After transformations on the right side of (14), we obtain:

$$\Delta V= k_1\cdot\left[\varepsilon^2/\left(\sqrt[m]{Q}-\varepsilon\right)\right]. \quad (15)$$

As proportionality coefficient k_1 as well as the expressions in the numerator and denominator (15) are non-negative, the following inequality is true: $\Delta V \geq 0$. This implies (12). Q.E.D.

Using Theorem 1 and equality (2), the mathematical model of the problem of minimizing the time of modular enumeration can be represented as:

$$\begin{cases} T = k_2\cdot m\cdot\sqrt[m]{Q} + k_3\cdot m\cdot Q \rightarrow \min; & (16) \\ V = k_1\cdot m\cdot\sqrt[m]{Q}; & (17) \\ V \leq \mathcal{V}, & (18) \end{cases}$$

where k_i (i=1,..3) are the proportionality coefficients, $\mathcal{V}$ is the amount of free RAM in the computer used.

Solving system (16) - (18) theorem is presented below.

Theorem 2**.** The number of equal size modules m, minimizing the value of upper bound of problem (16) - (18) goal function, coincides with the minimal integer value of m, satisfying (17), (18).

Proof. From (17) follows: $m\cdot\sqrt[m]{Q} = V/k_1$. (19)

We study two cases of system (16)-(18) upper bound of goal function value determination: in the first one for any integer m>1 true is inequality $V \leq \mathcal{V}$, whereas in the second case this inequality is true only for a subset of integer m values.

1. Let us suppose that true is the following expression: $\forall m \geq 2,\ k_1\cdot m\cdot\sqrt[m]{Q} \leq \mathcal{V}$. (20) Substituting the the right sides of (19) and (20) in

(16) we can get the expression for τ – the upper bound of T:

$$\tau = k_2 \cdot \mathcal{V}/k_1 + k_3 \cdot m \cdot Q. \quad (21)$$

As, according to the definitions above, in (21) values of $\mathcal{V}$, Q and of all the proportionality coefficients are nonnegative constants, value of $d\tau/dm$ is also nonnegative. It means, that minimal value of τ is determined by the minimal value of m, satisfying (20): in this case m=2.

2. Let us now suppose that instead of (20) being true the following conditions: $\forall m>2$, $\forall m>i\geq 0$:

a) $k_1 \cdot (m+i) \cdot \sqrt[m+i]{Q} \leq \mathcal{V}$; (22)

b) $k_1 \cdot (m-i) \cdot \sqrt[m-i]{Q} \geq \mathcal{V}$. (23)

If m' is the root of equation $k_1 \cdot m \cdot \sqrt[m]{Q} = \mathcal{V}$, satisfying conditions (22) and (23), then τ – the upper bound of modular enumeration time looks as follows:

$$\tau = k_2 \cdot \mathcal{V}/k_1 + k_3 \cdot m' \cdot Q. \quad (24)$$

Repetition of speculations above permits us to prove, that optimal value of m, minimizing the upper bound of T, is equal to m'.

Thus, the number of equal size modules m, minimizing the value of upper bound of modular enumeration running time, coincides with minimal integer value of m, satisfying (17), (18). Q.E.D.

The results above are illustrated by two examples: by an example1 above of using modular enumeration for integral (3) computing, and by the example 2 below of using modular enumeration to solve the knapsack problem, where m=3 is the root of equation $k_1 \cdot m \cdot \sqrt[m]{Q} = \mathcal{V}$, satisfying conditions (22) and (23).

Example 2**.**

Using m=3 modules of modular enumeration the following knapsack problem is solved:

$$\begin{cases} F=2z_1+7z_2+4z_3+6z_4+3z_5+8z_6 \rightarrow \max \\ R=9z_1+3z_2+8z_3+5z_4+7z_5+4z_6 \leq 10; \quad (25) \\ \forall i,\ z_i=1,0. \end{cases}$$

1. Results of the first two steps of the first stage of modular enumeration are presented bellow in the Tables 1-3: each i-th table corresponds to the i-th module m_i (i=1,2,3).

Table 1

m_1			
z_1	z_2	$F(m_1)$	$R(m_1)$
0	0	0	0
0	1	7	3
1	0	2	9
1	1	-∞	12

Table 2

m_2			
z_3	z_4	$F(m_2)$	$R(m_2)$
0	0	0	0
0	1	6	5
1	0	4	8
1	1	-∞	13

Table 3

m_3			
z_5	z_6	$F(m_3)$	$R(m_3)$
0	0	0	0
0	1	8	4
1	0	3	6
1	1	-∞	11

2 The "unpromising" components of each i-th module are presented by the vectors belonging to the third and fourth lines of Table i, $1 \leq i \leq 3$, since the vector in the second line of each module corresponds to the values F and R, which are simultaneously "better" than those located below: the value F is greater, and the value R is less. After elimination of un-promising vectors of variables, Tables 1-3 are below transformed into Tables 4-6, each i-th Table (i=4,5,6) corresponding to module m_{i-3}:

Table 4

m_1			
z_1	z_2	$F(m_1)$	$R(m_1)$
0	0	0	0
0	1	7	3

Table 5

m_2			
z_3	z_4	$F(m_2)$	$R(m_2)$
0	0	0	0
0	1	6	5

Table 6

m_3			
z_5	z_6	$F(m_3)$	$R(m_3)$
0	0	0	0
0	1	8	4

3 The Table 7 below illustrates the first step of the second stage of modular enumeration: each line contains different vectors of all modules presented in the Table 2 and the corresponding values of the system's (25) objective function F and the sum R of the components on the left side of this system's inequality.

Table 7

#	z_1	z_2	z_3	z_4	z_5	z_6	F	R
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	8	4
3	0	0	0	1	0	0	6	5
4	0	0	0	1	0	1	14	9
5	0	1	0	0	0	0	7	3
6	0	1	0	0	0	1	15	7
7	0	1	0	1	0	0	13	8
8	0	1	0	1	0	1	-∞	12

4 It is easy to see, that the optimal problem (25) solution is presented by the 6-th vector of variables in the Table 7. The gain η in running time when applying modular enumeration is compared with the brute force usage includes the following factors:

a) the number of arithmetical operations when calculating the values of F and R during the first step of the second stage of modular enumeration was two times less than in the case of the brute force;

b) due to the elimination of “unpromising” vectors of variables, the number of analyzed by modular enumeration different vectors of variables (Table 3 above) was eight times less than their total number in (25);

c) the latter does not guarantee a gain in time to find a solution compared to exhaustive modular search due to the time spent for searching and removing unpromising vectors of variables.

Statistical results of experimental verification of the effectiveness of knapsack problem solving by modular enumeration are presented in [5, 17-20].

5 Conclusions

1. The efficiency of modular enumeration is a predicted value, which increases with increasing search volume, regardless of the specifics of the problem being solved.

2. Optimal modular enumeration operating mode minimizing the value of upper bound of this procedure running time is characterized by:

   • a uniform distribution of components of enumerated objects among modules;

   • the number of equal size modules, coinciding with their minimal integer value, satisfying the free RAM restrictions.

Further development of modular enumeration can be associated with the following:

- shortening the running time for the first stage of this procedure i. e. preparation for the enumeration minimizing its repetitive computations by applying modular enumeration also for each module;
- verification of the effectiveness of composite algorithms using modular enumeration in relation to not yet investigated functions and the extension of this approach to new subject areas.

References

[1]. Johnson's algorithm: A key to solve optimally or approximately flow shop scheduling problems with unavailability periods. International Journal of Production Economics. Volume 121, Issue 1, September 2009, pp. 81-87.

[2]. Pettie, Seth; Ramachandran, Vijaya, An optimal minimum spanning tree algorithm, Journal of the ACM, 49 (1), 2002, pp.16–34.

[3]. Rego, César; Gamboa, Dorabela; Glover, Fred; Osterman, Colin, Traveling salesman problem heuristics: leading methods, implementations and latest advances, European Journal of Operational Research, 211 (3),2011, pp. 427–441.

[4]. Richard Durbin, David Willshaw. An analogue approach to the travelling salesman problem using an elastic net method (англ.) // Nature, Vol. 326, iss. 6114, 1987, pp. 689–691.

[5]. H. Ellis, S. Sartaj: Computing partitions with applications to the knapsack problem Journal of the Association for Computing Machinery, 21 (2), 1974, pp. 277–292.

[6]. V. O. Groppen and A. A. Berko, "Modular Technology of Definite Multiple Integrals Calculation: Analytical Analysis and Experimental Verification of Efficiency," Proceedings of the International Russian Smart Industry Conference (SmartIndustryCon), Sochi, Russian Federation, 2023, pp. 100-104.

[7]. Hammer, P. L.; Johnson, E. L.; Korte, B. H. , "Conclusive remarks", Discrete Optimization II, Annals of Discrete Mathematics, vol. 5, Elsevier, 2000, pp. 427–453.

[8]. Meryem Bouras & Abdellah Idrissi. A Survey of Parallel Computing: Challenges, Methods and Directions, Chapter in the Modern Artificial Intelligence and Data Science, 2023, pp 67–81.

[9]. Paweł Czarnul, Jerzy Proficz, and Krzysztof Drypczewski. Survey of Methodologies, Approaches, and Challenges in Parallel Programming Using High-Performance Computing Systems. In Methodologies for Highly Scalable and Parallel Scientific Programming on High Performance Computing Platforms, 2020, Article ID 4176794 https://doi.org/10.1155/2020/4176794

[10]. H. Land & A. G. Doig, An automatic method of solving discrete programming problems, Econometrica, Vol. 28, No. 3, 1960, pp. 497-520.

[11]. C. A. Brown and P. W. Purdom Jr.: An empirical comparison of backtracking algorithms. IEEE PAMI, 1982, pp. 309-315.

[12]. R. Bellman: On The Theory of Dynamic Programming. Proceedings of the National Academy of Sciences of the United States of America, Vol. 38, No. 8, 1952, pp. 716-719.

[13]. D. R. Morrison: New methods for branch-and-bound algorithms. Dissertation submitted for the degree of Doctor of Philosophy in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2014.

[14]. V. O. Groppen, and A. A. Berko: Composite version of B&B algorithm: experimental verification of the efficiency. J. Phys.: Conf. Ser. 1278 012029, 2019, pp. 1-8.

[15]. V.O. Groppen: Composite Versions of Implicit Search Algorithms for Mobile Computing. Proceedings of the Future Technologies Conference (FTC) 2020, Volume 2, November 5–6, 2020, pp. 336 – 348.

[16]. V.O. Groppen: Analysis of the Effectiveness of Composite Versions of Backtracking Algorithms. Conference proceedings, RusAutoConf 2020: Available: Advances in Automation II, Springer, 2021, pp 235-244.

[17]. V.O. Groppen, A New Family of Algorithms Searching Optimal Solutions to Discrete Optimization Problems. International journal of computers, Volume 15, 2021, pp 156 – 160.

[18]. V.O. Groppen: A new method for searching globally optimal solutions to the problems of discrete programming. Proceedings of the III All-Russian (national) scientific-practical conference with international participation, 2021, pp. 82 – 85. (russ).

[19]. V.O. Groppen: A New Method Searching Globally Optimal Solutions to Discrete Optimization Problems. Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3, pp 486 – 494.

[20]. V. O. Groppen & A.A. Berko. Modular Enumeration Algorithms: Analytical Study and Experimental Verification of Effectiveness. Lecture Notes in Networks and Systems, 2023, pp. 769–779.

[21]. L. Caccetta, A. Kulanoot: Computational Aspects of Hard Knapsack Problems. Nonlinear Analysis, 47, 2001, pp. 5547–5558.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflict of interest to declare that is relevant to the content of this article.