ISSN: 2994-9459

Research Article

Current Research in Statistics & Mathematics

Partitions, the P Versus NP Problem and Applications

Vassilly Voinov*

Independent scholar, Almaty, Republic of Kazakhstan

***Corresponding Author**
Vassilly Voinov, Independent scholar, Almaty, Republic of Kazakhstan.

Submitted: 2025, Apr 07; **Accepted:** 2025, May 07; **Published:** 2025, May 21

Citation: Voinov, V. (2025). Partitions, the P Versus NP Problem and Applications. *Curr Res Stat Math, 4*(2), 01-13.

Abstract

A new deterministic polynomial-time algorithm for sets of different positive integers partitioning has been introduced. The most important properties of the algorithm are that its time-complexity is $O(nlogn)$ and that it is invariant w.r.t. the number of elements in a particular part. An implementation of this algorithm and the trivial combinatorial show that the classical Partition and the 4-Partition problem, currently considered to be in the NP-complete and the NP-complete in the strong sense complexity classes, respectively, are both solvable in polynomial time, thus belonging to class P. These results permit to introduce the P-complete class of problems reducible to each other by a polynomial transformation. Since it is a subclass of the NP-complete complexity class, from the well-established theory of NP-completeness, it immediately follows that P = NP. This intriguing research can be characterized as a constructive theoretical, supported by Monte Carlo simulations, proof of the fundamental equality P = NP. The presented results permit us to develop the most efficient deterministic algorithms for solving numerous practical problems in discrete mathematics, business, management, industry etc.

Keywords: Integer Linear Programming, Combinatorics, Partitions, Polynomial Time Algorithms, Public-Key Cryptography, P =Np

1. Introduction

The most formal and strong definition of the P versus NP problem was formulated by Cook [2]. The key terms of the definition are: a) *A language* over a finite alphabet Σ is a subset L of the set Σ^* of finite strings over Σ, b) the *Turing machine M* (a formal analog of an algorithm) that accepts a string $w \in \Sigma^*$, c) the *language L accepted by M*

$L(M) = \{w \in \Sigma * | M \text{ accepts } w\}$.

The complexity class P of languages is defined by

P = $\{L|L = L(M)$ *for some deterministic M* (DTM) *running in polynomial time*$\}$.

The complexity class NP is

NP = $\{L|L = L(M)$ *for some nondeterministic M* (NTM) *running in polynomial time*$\}$.

If the terms were as above, then the definition would be as: "The P versus NP problem is to determine whether every language accepted by some nondeterministic algorithm in polynomial time is also accepted by some (deterministic) algorithm in polynomial time" (Cook, p. 1). A lot of researchers have tried to solve the problem during the last 54 years. Woeginger listed 116 printed sources [24]. 61 sources support the conclusion that P = NP, and 52 of them are in favor of P ≠ NP. It is worth also mentioning three recent theoretical attempts to prove that P ≠ NP . Combining the fact that the states of an NTM can be a power set of the corresponding DTM with the Cantor's theorem, Yang showed that an NTM is not equipotent to a DTM [25]. He concluded that "generating the power set $P(A)$ of a set A" is a non-canonical example to support that P ≠ NP. Kyritsis used the Zermelo-Frankel set theory [7]. He also thought that his "proof" is justified by the fact that the well-known RSA algorithm is safe. Today we know that the prime factorization problem is in P (Voinov),and hence, the RSA algorithm is unsafe [12,18]. Sing Kuang Tan used the Markov random field and Boolean algebra simplification [13]. He showed that "the Boolean algebra cannot be factorized into another Boolean algebra that can be solved in polynomial time (NP ≠ P)".

It is of interest also to mention several recent "proves" that P = NP. Zeilenberg used "3000 hours of CPU time on a CRAY machine" and presented "a "polynomial" time algorithm for the NP-complete subset sum problem" [26]. Unfortunately, this result is not convincing, because the time complexity of that

algorithm $O(n^{10^{10000}})$ is too high. Wen-Qi reduced "the undirected Hamiltonian cycle problem into the TSP problem with cost 0 or 1" and developed "an effective algorithm to compute the optimal tour of the transformed TSP" [23]. LaPlante proposed a polynomial time algorithm for solving clique problems. Panyukov considered the "Hamiltonian complement of the graph $G = (V(G), E(G))$" [8,10]. Using the linear programming, he "proves" the theorem that "all problems of NP class are polynomial-solvable with deterministic machines". The mathematical background and algorithms are discussed in Section 2. Section 2.3 is devoted to the classical Partition problem and its applications. The 4-Partition problem's solution is provided in Section 3. A discussion and conclusions of the research are presented in Section 4. R-scripts developed and used in this research are given in three appendices.

2. Mathematical Background

2.1 Power Series

Voinov and Nikulin considered a formal identity for the power series

$$\left(\sum_{l=0}^{m} b_l\, z^l\right)^{\alpha} = \sum_{k=0}^{\infty} a_k\,(\alpha, m) z^k, \tag{1}$$

where $\alpha \in \mathbb{C}$ and $b_l \in \mathbb{Z}$.They have proved the following

Theorem 1 [14]: *If power series in (1) exist, then*

$$a_k(\alpha, m) = \sum_{l_1=0}^{\left[\frac{(m-1)k}{m}\right]} \sum_{l_2=(2l_1-k)_+}^{\left[\frac{(m-2)l_1}{m-1}\right]} \cdots \sum_{l_{m-1}=(2l_{m-2}-l_{m-3})_+}^{\left[\frac{l_{m-2}}{2}\right]} \frac{(\alpha)_{k-l_1} b_0^{\alpha-k+l_1} b_1^{k-2l_1+l_2} \cdots b_{m-1}^{l_{m-2}-2l_{m-1}} b_m^{l_{m-1}}}{(k-2l_1+l_2)!\cdots(l_{m-2}-2l_{m-1})! l_{m-1}!}, \tag{2}$$

where $l_0 = k$, $l_m = 0$, $a_+ = \max\{0, a\}$, $(\alpha)_s = \alpha(\alpha-1)\cdots(\alpha-s+1)$ *with* $(\alpha)_0 = 1$, *and* $[x]$ *is the integer part of* x.

This result was used for deriving closed expressions for the Gauss and Bell polynomials, Bernoulli, Euler, Fibonacci, modified Stirling and C numbers.

The formula in (2) is not trivial. Consider, for example, the well-known Multinomial Theorem

$$(x_1 + x_2 + \cdots + x_m)^n = \sum_{\substack{k_1+k_2+\cdots+k_m=n \\ k_1,\ldots,k_m \geq 0}} \frac{n!}{k_1! k_2! \cdots k_m!} \prod_{t=1}^{m} x_t^{k_t}, \tag{3}$$

where k_i are nonnegative integers and $m, n \in \mathbb{N}$. If $\alpha = n$, then the formula in (2) produces the alternate closed form expression for the multinomial coefficients. If, e.g., $\alpha = 3$, $m = 2$, then the terms of the $(b_0 + b_1 + b_2)^3$ expansion are

$$a_k(3,2) = \sum_{l_1=0}^{\left[\frac{k}{2}\right]} \frac{(3)_{k-l_1} b_0^{3-k+l_1} b_1^{k-2l_1} b_2^{l_1}}{l_1!(k-2l_1)!}, 0 \leq k \leq 6. \tag{4}$$

Direct calculations produce the same, if using (3), 10 terms of the expansion

$$(b_o + b_1 + b_2)^3 = b_0^3 + 3b_0^2 b_1 + 3b_0 b_1^2 + 3b_0^2 b_2 + b_1^3 + 6b_0 b_1 b_2 + 3b_1^2 b_2 + 3b_0 b_2^2 + 3b_1 b_2^2 + b_2^3,$$

but the expression in (4) is in some sense simpler, because it does not need to enumerate nonnegative integer solutions of the equation $k_1 + k_2 + k_3 = 3$.

2.2 Partitions and Subset Sums

The Theorem 1 can also be used for solving subset sum problems, because it permits us to formulate the following

Theorem 2 *If a vector* $\boldsymbol{A} = (a_1, a_2, \ldots, a_n)^T$ *consists of elements* $a_i \in \mathbb{Z}^+$ *such that* $a_1 < a_2 < \cdots < a_n$, $n \geq 2$, *then the number* R_a *of* $\boldsymbol{A}$*'s partitions with exactly* $M \leq n$ *distinct elements summed to* $B \in \mathbb{N}$ *is*

$$R_a = \sum_{s_1=0}^{min\left(1,\left[\frac{B-Ma_1}{a_n-a_1}\right]\right)} \sum_{s_2=0}^{min\left(1,\left[\frac{B-Ma_1-s_1(a_n-a_1)}{a_{n-1}-a_1}\right]\right)} \dots \sum_{s_{n-2}=0}^{min\left(1,\left[\frac{B-Ma_1-s_1(a_n-a_1)-\cdots-s_{n-3}(a_4-a_1)}{a_3-a_1}\right]\right)} 1, \quad (5)$$

where s_i, $i = 1, \dots, n$, $\in \{0,1\}$, $s_1 + s_2 + \cdots + s_{n-2} \le M$, $s_{n-1} = (B - Ma_1 - s_1(a_l - a_1) - \cdots - s_{n-2}(a_3 - a_1))/(a_2 - a_1)$, *and* $s_n = M - s_1 - s_2 - \cdots - s_{n-1}$. *In other cases* R_a *is zero. All existing partitions or, what is the same, the corresponding 0-1 solutions of the equation*

$$a_1 s_1 + a_2 s_2 + \cdots + a_n s_n = B \quad (6)$$

can be enumerated as

$$\{a_1^{s_n}, a_2^{s_{n-1}}, \dots, a_n^{s_1}\}, \quad (7)$$

where sets $\{s_1, s_2, \dots, s_{n-2}\}$ *are defined by the summation indices in (5). The notion (7) means that in a particular solution there will be* s_n *terms* a_1, s_{n-1} *terms will be* a_2, *and so on.*

Proof Using the Theorem 1, Voinov and Nikulin [15] derived the formula for the number R_a of the equation (6) nonnegative integer solutions with exactly M parts as:

$$R_a = \sum_{s_1=0}^{\left[\frac{B-Ma_1}{a_n-a_1}\right]} \sum_{s_2=0}^{\left[\frac{B-Ma_1-s_1(a_n-a_1)}{a_{n-1}-a_1}\right]} \dots \sum_{s_{n-2}=0}^{\left[\frac{B-Ma_1-s_1(a_n-a_1)-\cdots-s_{n-3}(a_4-a_1)}{a_3-a_1}\right]} 1, \quad (8)$$

where s_i, $i = 1, \dots, n$, are nonnegative integers, $s_{n-1} = \frac{B-Ma_1-s_1(a_n-a_1)-\cdots-s_{n-2}(a_3-a_1)}{a_2-a_1}$, $s_1 + s_2 + \cdots + s_{n-1} \le M$, and $s_n = M - s_1 - s_2 - \cdots - s_{n-1}$. If conditions are not satisfied, then $R_a = 0$. The existing solutions are defined by (7).

The solutions or partitions defined by (8) may have several identical terms a_i, $i = 1, \dots, n$. If, e.g., $\boldsymbol{A} = (1,2,3)^T$, $B = 5$, $M = 3$, $n = 3$, then (8) gives two partitions: $\{1^1, 2^2, 3^0\} = 1 + 2 + 2 = 5$ and $\{1^2, 2^0, 3^1\} = 1 + 1 + 3 = 5$. To adjust (8) for solving the subset sum problem, one has to replace the upper limits of summation by the expressions that take only values 0 and 1 as required. Namely:

$\left[\frac{B-Ma_1}{a_n-a_1}\right]$ by $min\left(1, \left[\frac{B-Ma_1}{a_n-a_1}\right]\right)$, $\left[\frac{B-Ma_1-s_1(a_n-a_1)}{a_{n-1}-a_1}\right]$ by $min\left(1, \left[\frac{B-Ma_1-s_1(a_n-a_1)}{a_{n-1}-a_1}\right]\right)$, and so on.

Corollary 1 *Consider the following numerical example with* $\boldsymbol{A} = (127, 131, 153, 175, 194, 220)^T$. *Let* $M = 3$, $B = 500$. *Using Theorem 2, find partitions of his set with exactly 3 parts that are summed to* $B = 500$. *Using (5) and required conditions, one gets*

$$R_a = \sum_{s_1=0}^{1} \sum_{s_2=0}^{min\left(1,\left[\frac{119-93s_1}{67}\right]\right)} \sum_{s_3=0}^{min\left(1,\left[\frac{119-93s_1-67s_2}{48}\right]\right)} \sum_{s_4=0}^{min\left(1,\left[\frac{119-93s_1-67s_2-48s_3}{26}\right]\right)} 1, \quad (9)$$

$$s_5 = \frac{119-93s_1-67s_2-48s_3-26s_4}{4}, \text{ and } s_6 = 3 - s_1 - s_2 - s_3 - s_4 - s_5.$$

Calculating the binary branching process in (9), s_5 and s_6, one obtains sets {0, 1, 1, 0} and {1, 0, 0, 1} of the summation indices $\{s_1, s_2, s_3, s_4\}$ and two pairs {1, 0} and {0, 1} for $\{s_5, s_6\}$ respectively. The two defined by (7) solutions of the equation

$$127s_1 + 131s_2 + 153s_3 + 175s_4 + 194s_5 + 220s_6 = 500 \quad (10)$$

are {131, 175, 194} and {127, 153, 220}. These calculations present actually the balanced binary search tree provided in Figure 1.

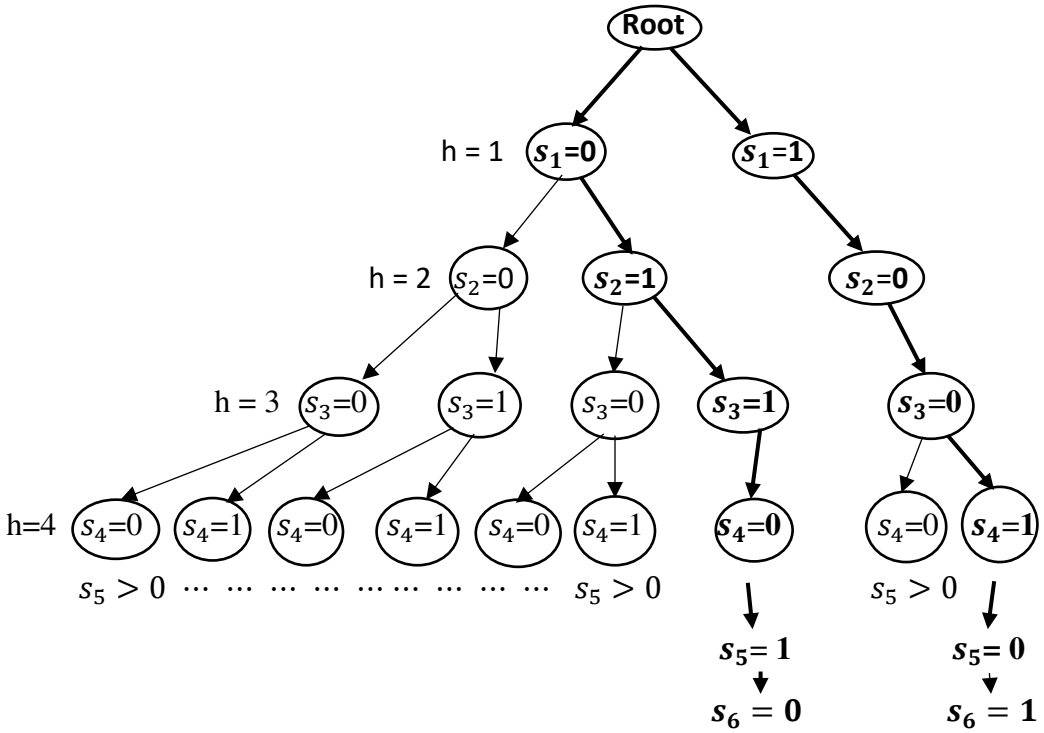

Figure 1: The binary tree generated by the algorithm in (9)

From Fig.1 one sees that seven leaves of the tree give positive values for s_5 instead of 0 or 1 as required, and, hence, paths ending by those leaves cannot be solutions. At the same time, the rest two leaves give $s_5 = 1$, $s_6 = 0$ and, respectively, $s_5 = 0$, $s_6 = 1$. Thus, bold-faced arrows and nodes link to s_5 and s_6 show two binary solutions' representations {0,1,1,0,1,0} and {1,0,0,1,0,1} of partitions {131,175,194} and {127,153, 220}.

Analogous (as a tree) presentations of the algorithm in (5) permit to define its time-complexity as follows. The first step of the algorithm presents the sequence of $n - 2$ embedding sums from 0 to 1 as a balanced binary search tree. The height h of the tree equals $n - 2$. From the theory of binary search trees (see, e.g., https// www.geeksforgeeks.org/binary-tree) it follows that: 1) The worst case number N of a tree nodes equals $2^{h+1} - 1$, 2) One search takes $O(logN)$ time (consult Adel'son-Vel'skii and Landis [1]). Since $logN = log(2^{h+1} - 1) < n - 1$, where n is the dimension of a vector $\boldsymbol{A}$, then the complexity of the algorithm's first step is $O(logn)$. The second step of the algorithm is a linear search of size $O(n)$ in the list of the last level leaf nodes equals 0 or 1. Thus, the time complexity of the entire algorithm is $O(nlogn) < O(n^2)$, because $xlogx < x^2$ for any $x \in \mathbb{R}$, if $x > 1$. In the sequel, we shall denote this deterministic algorithm as PSA (power series algorithm). It is of importance to note that the PSA complexity, on the contrary to the simple intuitive combinatorial algorithm (ICA) used in Voinov [17], does not depend on M.

Corollary 2 *Consider the zero-one integer programming problem that is considered to be NPcomplete (Garey and Johnson [4], p. 245). The problem means finding all solutions to the equation (6). To do this, the Theorem 2 suggests using all partitions with M = 1,2, … , n parts sequentially.*

Since the complexity of a particular partition does not depend on M and is $O(nlogn)$, then the entire solution will have the complexity of $O(n^2logn) < O(n^3)$. Thus, the 0-1 integer programming problem is in P. The excellent fit (R^2 = 0.9991) of experimental data (Mt1 from Table 1 of Voinov [17], p.4) does not contradict this conclusion. Moreover, Fig.1 of that paper provides a Monte Carlo simulated argument in favor of the inequality $O(n^2logn) < O(n^3)$.

The time-complexity of the two-step ICA is defined as follows: 1) the first step, which is a construction $\binom{n}{M}$ combinations of a vector $\boldsymbol{A}$ using the lexicographic of complexity $O(n^M)$ algorithm of Nijenhuis and Wilf [9], 2) the second step is the linear combinations' search with the complexity $O(n)$. Thus, the entire complexity of the ICA is $O(n^M)$. Probably, this complexity can be improved up to $O(n^2)$ using, e.g., the ideas of Genitrini and Pépin [5].

From the above theory, it follows that for any $\boldsymbol{A}$ both the PSA and ICA produce same solutions.

2.3 Partitions, Algorithms' Validity and Applications

Both the PSA and ICA have been realized as the R-script "2-PartsVG3" (see Appendix 1) based on the R-package "nilde" [11]. This script, on the contrary to Theorem 2, uses the decreasing order of a vector $\boldsymbol{A} = (a_1, a_2, \dots, a_n)^T$. Since solutions of the equation in (6) are invariant w.r.t. the order of summands, one will get the

same partitions as in (7). This permits us to compare the numerical results of the research with those in Voinov [17].

The partition problem that was defined by Garey and Johnson [4], p.90, can be formulated as follows:
Instance: Let the set $\boldsymbol{A} = \{a_1, a_2, \dots, a_n\}$ and the sizes $s(a_1), s(a_2), \dots, s(a_n)$ in $\mathbb{Z}^+$ form an arbitrary instance for Partition. Define $\boldsymbol{B'} = \Sigma_{a\in\boldsymbol{A}}\, s(a)$.

Question: Can A be partitioned into two disjoint subsets $\boldsymbol{A'}$ and $\boldsymbol{A} - \boldsymbol{A'}$ such that

$$\sum_{a\in\boldsymbol{A'}} s(a) = \sum_{a\in\boldsymbol{A}-\boldsymbol{A'}} s(a). \qquad (11)$$

Garey and Johnson [4], p.91, provided an instance constructed by a "dynamic programming like" algorithm and showed that it is not polynomial in time. From this they erroneously decided that the Partition problem is in NP. Their conclusion would be correct, if they proved that polynomialtime algorithms were impossible for the Partition problem.

Using the generating function algorithm (GFA) (suggested in Voinov and Nikulin [16]), Voinov [17] has shown that the GFA answers "yes" to the **Question**. The same answer is obtained using the PSA and ICA with the help of script "2-PartsVG3".

With $k = 1000, M = 2, n = 6, B = 50, g\$p. n = 2$ the script produces 9 pairs of disjoint sets $\boldsymbol{A'}$ and $\boldsymbol{A} - \boldsymbol{A'}$ answering "yes" on the **Question**. Two of them are given for the illustration: 1) {33,17} and {29,21} for $\boldsymbol{A} = (36,33,31,29,21,17)T$ ($\boldsymbol{B'} = 167$), 2) {31,19} and {28,22} for $\boldsymbol{A} = (45,31,29,28,22,19)^T$ ($\boldsymbol{B'} = 174$). Note that the answers do not depend on the divisibility of $\boldsymbol{B'} = \Sigma_{a\in\boldsymbol{A}}\, s(a)$ by 2. These counterexamples disprove the opinion of Garey and Johnson [4], p.90, that "if $\boldsymbol{B'}$ is not evenly divisible by 2, then we know that no subset $\boldsymbol{A'} \subseteq \boldsymbol{A}$ can possibly satisfy (11), so we can immediately respond "no" for this instance". It is of importance to note that this result does not depend on the quality of pseudo-random numbers used by the script, because the instances of interest really exist and are reproducible. An interested reader may generate his/her own instances using the script "2-PartsVG3" with any seed() and desired parameters M, m, B.

The numerical examples, which answer "yes" to the **Question**, solve the partition decision problem. To illustrate Theorem 2 and assess complexities of the PSA and ICA, consider the following experiment.

Exp. 1 For 10 pairs (n, B) with $n \in (4,6, \dots ,22)$ and $B \in (40,50, \dots ,130)$ 100,000 random samples of different positive integers uniformly distributed over the range (1, B) were generated using the R-command "sample (1:B, n, replace=FALSE)". An application of the R-package "microbenchmark" permits us to assess the computing time for solutions obtained by the PSA and ICA with the relative standard deviation of the mean $\delta < 1.2\%$. Results of the simulation for $M = 2$ are given in Table 1 and Figure 2.

n	B	$n+B+nB$	Mt1	Mt2
4	40	204	0.0004706	0.0001208
6	50	356	0.0007269	0.0001299
8	60	548	0.0013264	0.0001440
10	70	780	0.0025587	0.0001557
12	80	1052	0.0049015	0.0001722
14	90	1364	0.0095082	0.0001859
16	100	1716	0.0171800	0.0002012
18	110	2108	0.0309200	0.0002216
20	120	2540	0.0586660	0.0002462
22	130	3012	0.0982440	0.0002698

Mt2 - same for the ICA.

Table 1: Mt1 – mean time in seconds needed to enumerate existing solutions using the PSA, Mt2 - same for the ICA.

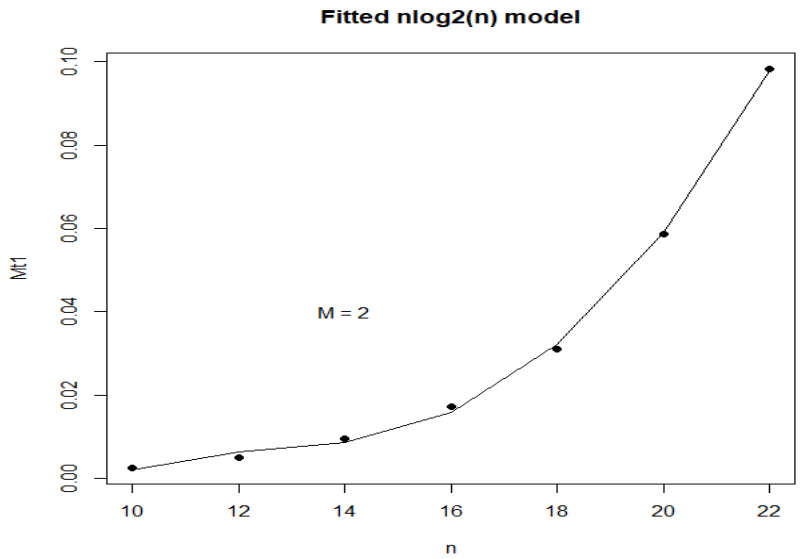

Figure 2: Fit of Mt1 by nlog2(n) model

Circles are values of Mt1, the solid line presents predicted by the R-command lm() values for the model. Statistical criteria of the fit are: residual standard error = 0.001434 on 3 degrees of freedom, multiple R-squared = 0.9992, adjusted R-squared = 0.9983. F-statistic =1203 on 3 and 3 DF (p-value: $4.06 \cdot 10^{-5}$), deviance = $6.17 \cdot 10^{-6}$, Akaike's information criterion AIC = −67.73. The numerical values of the criteria provide an excellent computing times' fit for the PSA that do not contradict the theory of Section 2.2. The Monte Carlo experiments of Voinov do not contradict the polynomial-time complexity of the ICA. Table 1 and Microsoft Excel fit for dependencies of the Mt2 on n and $n + B + nB$ confirm that conclusion, because Mt2 is polynomial in nB (consult Garey and Johnson, p.500) [3]. In view of the above, it is of interest to revisit Exp.3 of Voinov [17], p.10, Table 3. This experiment was based on the generating function algorithm (GFA) suggested by Voinov and Nikulin [16]. The GFA, like the PSA, solves subset sum problems using the embedding sums from 0 to 1, which are actually balanced binary search trees, and, hence, should be of $O(nlogn)$ complexity. Figure 3 below confirms this conclusion.

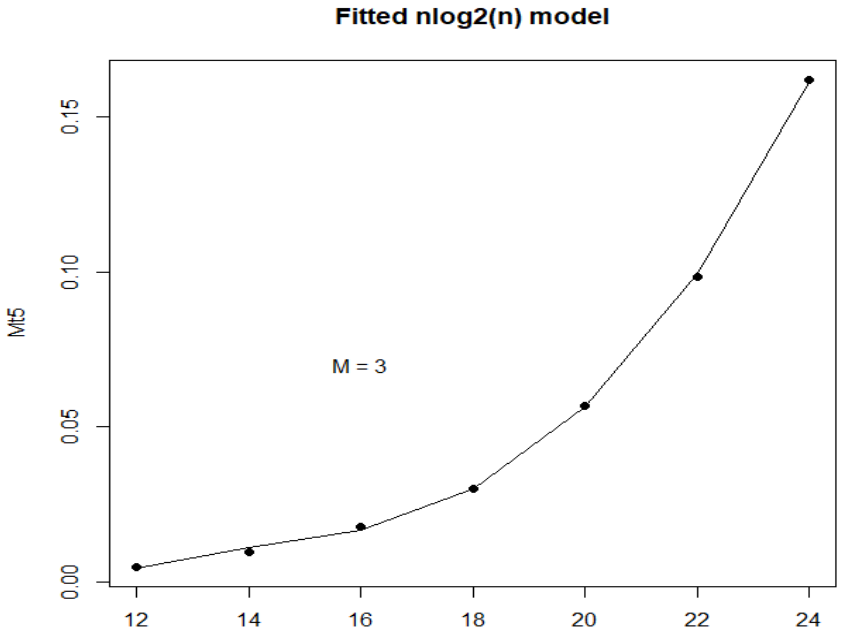

Figure 3: Fit of Mt5 (see Table 3 of Voinov [17]) by nlog2(n) model

Circles are values of Mt5; the solid line presents predicted values for the model. Statistical criteria for the fit are: residual standard error = 0.00154 on 3 degrees of freedom, multiple Rsquared = 0.9996, adjusted R-squared = 0.9993. F-statistic = 2813 on 3 and 3 DF (p-value: $1.14 \cdot 10^{-5}$), deviance = $7.074 \cdot 10^{-5}$, Akaike's information criterion AIC= −66.77.

Figures 2, 3 and the values of statistical criteria confirm the correctness and the time-complexity of the PSA and GFA. It is worth mentioning the following important properties of the PSA. Using data from Table 1 the Microsoft Excel gives the following fit with $R^2 = 0.9996$ for the dependence of Mt1 on $p = n + B + nB$ Mt1 $= 5 \cdot 10^{-12}p^3 - 5 \cdot 10p^2 + 5 \cdot 10^{-6}\,p$-0.0005, where the parameter p "reflects the number of symbols that would be required to describe the instance in a "reasonable" and "concise" manner". The polynomial $O(p^3)$ dependence of Mt1 on nB means that the complexity of the PSA stays polynomial even for extremely large input numbers. Cook [2], p.2, noted that "a constructive proof of P=NP" will lead to the potentially stunning practical consequences". One may consider the PSA as such proof. The PSA is exact and effective with the time-complexity bounded above by a polynomial. Actually, it is applicable to hundreds known as NP-complete problems, routing, sequencing, scheduling, planning, etc. Numerical exact deterministic working algorithms for all those problems can be based, e.g., on the PSA.

3.4 Partition

The 4-Partition problem is defined by Garey and Johnson, p.96, as follows:

Instance: A finite set $\boldsymbol{A}$ of $4m$ elements, a bound $B \in \mathbb{Z}^+$, and a "size" $s(a) \in \mathbb{Z}^+$ for every $a \in \boldsymbol{A}$ such that every $s(a)$ satisfies B/5< $s(a) < B/3$, and such that $\sum_{a \in A} s(a) = mB$.

Question: Can $\boldsymbol{A}$ be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ such that for $1 \le i \le m$ $\sum_{a \in S_i} s(a) = B$. (The above constraints on the item sizes imply that every such S_i will contain exactly four elements from $\boldsymbol{A}$).

This problem is of special interest because Garey and Johnson [4], pp.96-99, presented analytical proof that it is NP-complete in the strong sense. Any NP-complete problem is NP-complete in the strong sense if it is NP-complete even for its restricted version. Garey and Johnson, p.97, proved the theorem that 4-Partition is NP-complete in the strong sense. Their Theorem 4.3 states that "4-Partition is NP-complete even when restricted to instances I with $Max[I] \le 2^{16}|\boldsymbol{A}|4$, where $Max[I]$ to be max $\{s(a)\colon a \in \boldsymbol{A}\}$". They proved this theorem by presenting a polynomial transformation from the 3-dimensional matching problem to this restricted version of the 4-Partition. As in the case of Partition, it has to be noted that the conclusion about NP-completeness of the 4-Partition would be correct if the authors proved that no polynomial-time algorithms were possible for this problem. Consider the experimental 4-Partition decision problem solution. Using the R-script "2PartsVG2" for k = 5000, seed(852), and parameters (4,3,500), one will get 7 random instances answering "yes" to the question. Two of them are given below for the illustration.

```
[1] 165 151 148 142 130 119 115 112 110 105 102 101          [1]
156 147 144 141 140 120 118 117 106 105 104 102  #A
 [1] 1500                                                    [1]
1500   # mB
 [,1] [,2] [,3]                                  [,1] [,2]
[,3]   # solutions by ICA
[1,] 165  151  148                                           [1,]
156  147  141
[2,] 119  142  130                                           [2,]
120  144  140
[3,] 115  105  112                                           [3,]
118  105  117
[4,] 101  102  110                                           [4,]
106  104  102
```

Solutions by the PSA are absolutely the same as by ICA and, thus, are not shown here. To assess the time complexity of the ICA in this case, consider the following computer experiment:

Exp. 2. For every 14 pairs $[n, B]$ with $n \in$ [8,12, … ,60] and B∈ [45,60, … ,240] 35,000 samples were generated using the R-command "sample($R1$:$R2$,4m,replace=FALSE)", where $R1 = [B/5] + 1$ and $R2 = [B/3] - 1$. The computing times for partitions with the relative standard deviation of the mean $\delta < 0.2\%$ were obtained using the R-script "3-PartsVG91". Results of the simulation are provided in Table 2 and Figure 4.

m	n	B	p	Mt7, sec.	k
2	8	45	413	0.0001761	1000
3	12	60	792	0.0005113	1000
4	16	75	1291	0.0014972	1000
5	20	90	1910	0.0036885	1000
6	24	105	2649	0.0077785	1000
7	28	120	3508	0.0146864	1000
8	32	135	4487	0.0256434	1000
9	36	150	5586	0.0420777	4000
10	40	165	6805	0.0662237	4000
11	44	180	8144	0.0982870	4000
12	48	195	9603	0.1460729	4000
13	52	210	11182	0.2033357	4000
14	56	225	12881	0.2801219	4000
15	60	240	14700	0.3735920	4000

Table 2: Mt7 is the mean computing time to enumerate one 4-Partition instance using the ICA. k is the number of generated random samples of size $4m$

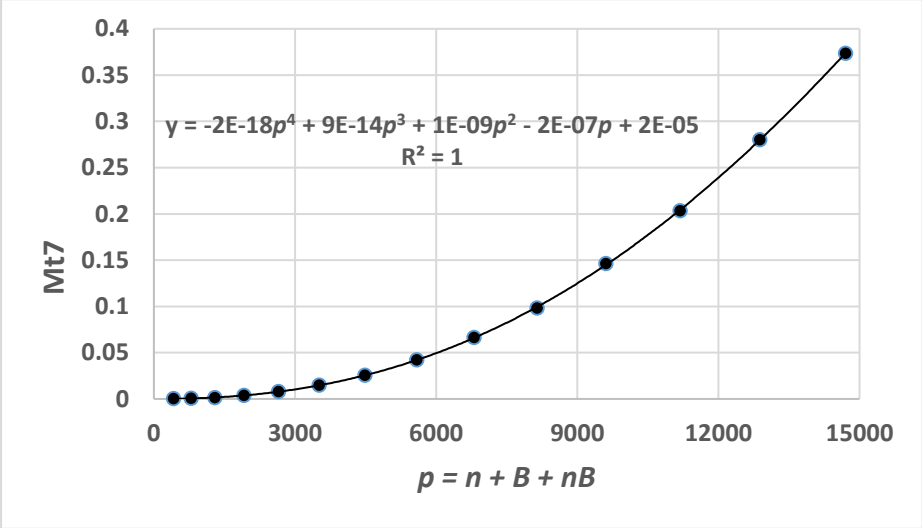

Figure 4: Dependence of Mt7 on $p = n + B + nB$. The trend line y with $R^2 = 1$ was obtained by the Microsoft Excel

The best fitted polynomial model $y = -2 \cdot 10^{-18}p^4 + 9 \cdot 10^{-14}p3 + 1 \cdot 10^{-9}p2 - 2 \cdot 10^{-7}p + 2 \cdot 10^{-5}$ possesses the following statistics: multiple and adjusted R^2 equal 1, the residual standard error is 0.0006631 on 9 degrees of freedom (DF), the p-value of the F-test on 4 and 9 DF is less than $2.2 \cdot 10^{-16}$, coefficients at p^2, p^3 and p^4 are significant with p-values 0.00163, 0.00322, and 0.01 respectively, and the Akaike's information criterion AIC= −159.38. The deterministic ICA that solves the 4-Partition problem is based actually on solving the equation (6), which is correctly encoded by a string of length $O(\log p)$, where $p = n + B + nB$ in the worst case. Thus, the above result shows that the 4-Partition problem, regardless of the value of input numbers, is solvable by ICA in polynomial time. To apply the PSA for the 4-Partition problem, consider the following computer experiment.

Exp. 3. For 6 pairs $[n, B]$ with $n \in [20, 24, \ ,40]$ and $B \in [800, 1000, \ldots, 1800]$ 19,400 samples were generated using the R-command "sample(R1:R2, n, replace=FALSE)", where $R^1 = [B/5] + 1$ and $R^2 = [B/3] - 1$. Results of the simulation are provided in Table 3 and Figure 5.

n	B	$p = n + B + nB$	Mt8, *sec.*	k
20	800	16,820	0.458586	10,000
24	1000	25,024	1.511360	5,000
28	1200	34,828	6.251800	2,000
32	1400	46,232	19.31787	2,000
36	1600	59,236	70.22397	200
40	1800	73,840	155.4354	200

Table 3: Mt8 is the mean computing time to enumerate all 4-Partition instances for every pair of [n, B] of size *n. k* **is the number of samples generated.**

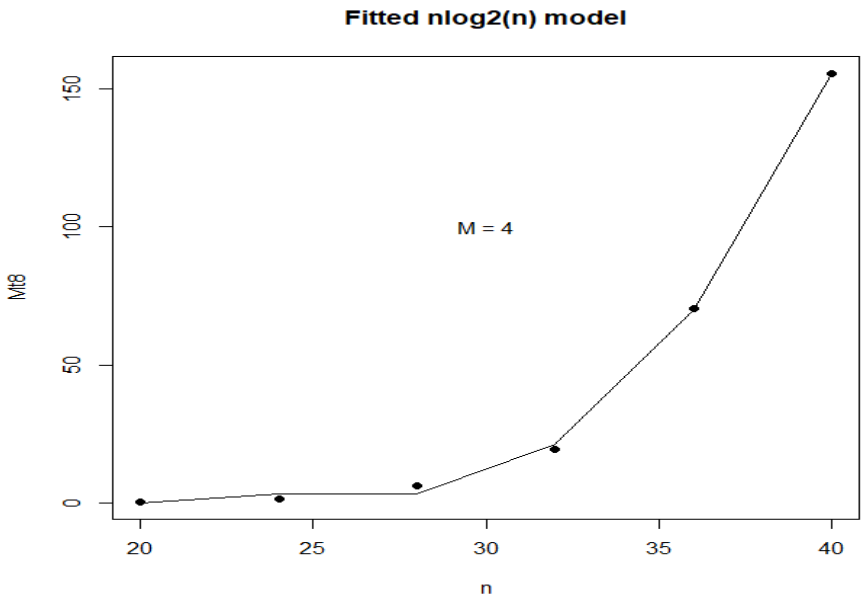

Figure 5: Fit of Mt8 by nlog2(n) model

Circles are simulated values of Mt8; the solid line presents predicted values for the model. Statistical criteria for the fit are: residual standard error = 2.993 on 2 degrees of freedom, multiple R-squared = 0.999, adjusted R-squared = 0.9976, F-statistic = 699.7 on 3 and 2 DF, p-value = 0.001427, AIC = 33.59, deviance = 17.916. From Fig. 5 one sees that the 4-Partition problem is solvable by the PSA with the time-complexity of $O(nlogn) < O(n^2)$, The polynomial-time solvability of the problem is also confirmed by the Mt8 fit on $p = n + B + nB$ provided in Figure 6.

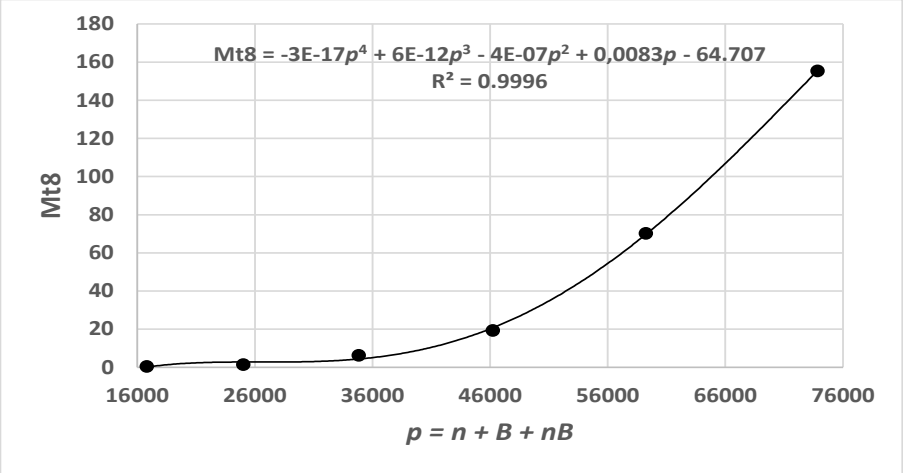

Figure 6: Fit of Mt8 by $p = n + B + nB$

From all the above one may conclude that both the theory of Section 2.2 and Exp.3 prove the 4-Partition problem's solvability in polynomial time by the PSA and the ICA and, thus, this problem belongs to class P. Let L denote a language. Cook [2], p.4, formulated and proved the following

Proposition

(a) If $L_1 \leq_p L_2$ and $L_2 \in$ P, then $L_1 \in$ P.

(b) If L_1 is NP-complete, $L_2 \in$ NP, and $L_1 \leq_p L_2$, then L_2 is NP-complete.

(c) If L is NP-complete and $L \in$ P, then P = NP.

Since the 4-Partition problem is due to Garey and Johnson, pp. 96-99, NP-complete and, at the same time, belongs to P, then from the item (c) of the Proposition, it immediately follows that P=NP.

4. Conclusions

A new polynomial-time algorithm for solving the subset sum problem has been described in Section 2.2. The algorithm's key point is the 0-1 solutions' enumeration for a linear Diophantine equation (6), which is a particular case of that equation to be solved in nonnegative integers. Two approaches for solving this problem are possible. The first one (Voinov and Nikulin is based on the formal identity for power series (see Theorem 1) [14]. The second approach (Voinov and Nikulin) uses generating functions [16]. It leads to the GFA. Both two approaches use, as a starting point, the famous binomial theorem. Up to now, only computer simulations were used to assess the time-complexity of those algorithms. It has been shown that the PSA represents balanced binary search tree. The theory of binary trees permits us to assess its complexity as $O(nlogn)$, which is less than the complexity $O(n^M)$, $M \geq 2$, of the trivial combinatorial algorithm ICA. The most important property of the PSA is that its complexity does not depend on M. These theoretical results, supported by Monte Carlo simulations, and the Cook's Proposition prove that P = NP. From this conclusion and the fact that P ⊆ NP it follows that the well-established and recognized by the research community NP-completeness theory is also valid for all P problems. In particular, it follows that all those problems are reducible to each other by polynomial transformations, known for NP-complete problems, thus forming the large P-complete in the sense of Karp class, which is a subclass of the NP-complete one [6]. This circumstance is very important for practitioners because it guarantees the possibility to develop efficient algorithms for solving problems of interest using any already known one for some P-complete problem (e.g. Partition). Such algorithms, correctness of which is confirmed by the theory of Section 2, have already been presented for solving several discrete optimization problems: zero-one integer programming (see Corollary 2 of Section 2.2), one-dimensional bin-packing, traveling salesman and tiling and storing [19-22].

In Section 1 it was noted that Kyritsis used the "safeness" of the RSA algorithm as an argument in favor of inequality P ≠ NP [7]. He thought that "safety" is guaranteed by the fact that the prime factorization problem is in NP. Today we know that this problem is actually in P [18] and, hence, the RSA algorithm being unsafe is in favor of equality P = NP. Currently, we cannot include the prime factorization problem in the P-complete class, since we do not know yet a polynomial-time transformation from any one problem in the NP-complete class to this problem. To summarize, one may say that this paper proposes a constructive theory supported by Monte Carlo simulations as proof of equality P = NP that permits us to develop the most efficient deterministic polynomial-time algorithms for solving numerous practical problems in discrete mathematics, business, management, industry etc.

Declarations

- Funding: Not applicable
- Conflict of Interest: Not applicable
- Ethics Approval and Consent to Participate: Not applicable
- Consent for Publication: Not applicable
- Data Availability: All data except Table 1 from Voinov [17] were obtained in this research
- Materials Availability: Not applicable
- Code Availability: R-scripts developed and used are given in appendices
- Author Contribution: Not available

Acknowledgements

The author is grateful to Hazell Mitchell for her enormous help in this article publishing.

References

1. Adel'son-Vel'skii, G. M. (1962). An algorithm for the organization of information. *Soviet Math., 3,* 1259-1263.
2. Cook, S.: The P versus NP problem. (2000). https://www.claymath.org/pvsnp.
3. Garey, M. R., & Johnson, D. S. (1978). ``strong"np-completeness results: Motivation, examples, and implications. *Journal of the ACM (JACM), 25*(3), 499-508.
4. Garey, M., D. Johnson, D.: (1979). Computers and Intractability: A Guide to the theory of NP- Completeness. Freeman W.H. and Co., New York.
5. Genitrini, A., & Pépin, M. (2021). Lexicographic unranking of combinations revisited. *Algorithms, 14*(3), 97.
6. Karp, R. M. (2009). Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art* (pp. 219-241). Berlin, Heidelberg: Springer Berlin Heidelberg.
7. Kyritsis, C. K. (2021). Review of the Solutions of the Clay Millennium Problem about P≠ NP= EXPTIME. *World Journal of Research and Review, 13*(3), 21-26.
8. LaPlante, M. (2015). A polynomial time algorithm for solving clique problems. *arXiv preprint arXiv:1503.04794*.
9. Nijenhuis, A., & Wilf, H. S. (2014). *Combinatorial algorithms: for computers and calculators*. Elsevier.
10. Panyukov, A. (2014). Polynomial solvability of $ NP $-complete problems. *arXiv preprint arXiv:1409.0375*.
11. Pya Arnqvist, N., Voinov, V., Makarov, R., Voinov, Y.:"nilde": Non-negative integer solutions of linear Diophantine equations with applications (2021). R-package version 1.1-7. https://CRAN, R-project.org/package-nilde, https://doi.org/10.13140/RG.2.2.26198.19523

12. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21*(2), 120-126.
13. Tan, S. K. (2021, July). *Prove Np not equal P using Markov Random Field and Boolean Algebra Simplification.*
14. Voinov, V., & Nikulin, M. (1995). Generating functions, problems of additive number theory, and some statistical applications. *Revue Roumaine de Mathematiques Pures et Appliquees, 40*(2), 107-148.
15. Voinov, V., Nikulin, M. (1995b). Corrigendum, Romanian journal of pure and applied mathematics 41(9-10), 713.
16. Voinov, V. G., & Nikulin, M. S. (1997). On a subset sum algorithm and its probabilistic and other applications.
17. Voinov, V. (2023). An experimental supported by some strong theoretical arguments proof of the fundamental equality P= NP. *Central Asia Business Journal.*
18. Voinov, V. (2023b). A simple enumerative polynomial-time algorithm for the prime factorization. *Central Asian Business Journal 23*(2), 1-6.
19. Voinov, V., Arnqvist, N. P., & Voinov, Y. (2018). Polynomial in time nonnegative integer solutions of knapsacks and similar problems in R: P= NP?. *Mathematical Journal, 18*(2), 47-58.
20. Voinov, V., Makarov, R., & Voinov, Y. (2019). An exact polynomial in time solution of the one-dimensional bin-packing problem. *Submitted to ASMDA, 2019*.
21. Voinov, V., & Pya Arnqvist, N. (2021). An exact polynomial-time algorithm for the optimal solution of traveling salesman problems. *Central Asia Business Journal, 12*(4), 17-32.
22. Voinov, V. (2022). An algorithm for deriving classical and Generalized Frobenius numbers: applications in tiling and storing. Central Asian Business Journal 13(1), 23-35.
23. Duan, W. Q. (2012). A constructive algorithm to prove P= NP. *arXiv preprint arXiv:1208.0542*.
24. Woeginger, G. J. (2016). P-versus-NP page. url: https://www. win. tue. nl/~ gwoegi. P-versus-NP. htm.(Retrieved September 13, 2021).
25. Yang, Z. (2011). A non-canonical example to support P is not equal to NP. Transactions of Tianjin University, 17, 446-449.
26. Zeilberger, D. (2023). *Research Announcement: A Computer-Generated Proof that P= NP*.

Appendix 1 The R-script "2-Parts_VG_3"

```
library(combinat);library(nilde);library(microbenchmark);options(warn=-1)
genSample<-function(M,m,B){
x<-array(0,dim=M*m);R1<-floor(B/(M+1))+1;R2<-floor(B/(M-1))-1
x<-sample(R1:R2,M*m,replace=FALSE);xx<-sort(x,decreasing=TRUE)
d2<-M*m; d4<-M; d6<-B;d1<-list(xx,d2,d4,d6,m)
return(d1);}
k<-1000
Tnat<-rep(NA,k);Tcomb<-rep(NA,k)
set.seed(8542); Tb<-proc.time()[3]
for(i in 1:k){
ns<-genSample(2,3,50)
d4<-ns[[3]]; d6<-ns[[4]];d7<-ns[[1]];d2<-ns[[2]];d8<-ns[[5]]
g<-get.subsetsum(a=d7,n=d6,M=d4,problem="subsetsum01")
if(g$p.n==2){; dv2<-rep(0,d2-1)
for(i1 in 1:(d2-1)){
dv2[i1]<-d7[1]-d7[i1+1];}
dv2<-c(dv2); a4<-d4*d7[1]-d6
T30<-microbenchmark(g<-get.subsetsum(a=dv2,n=a4,M=d4,problem="subsetsum01"),times=1L)
T301<-T30$time/1000000000;Tnat[i]<-T301; b10<-g$solutions
d81<-g$p.n;s0<-rep(0,d81)
for(j in 1:d81){
s0[j]<-d4-sum(b10[,j]);}
b11<-rbind(s0,b10)
row.names(b11)<-paste("s",1:d2,sep="")
print(d7); print(b11)
a2<-microbenchmark(a4<-combn(x=d7,m=d4,fun=NULL,simplify=TRUE),times=1L)
T1<-a2$time/1000000000; a3<-as.matrix(a4)
a5<-microbenchmark(Solcomb<-as.matrix(a3[,colSums(a3)==d6]),times=1L)
T2<-a5$time/1000000000; print(Solcomb)
Tcomb[i]<-T1+T2
```

```
};}
Te<-proc.time()[3]; print(Te-Tb)
Tnat<-Tnat[!is.na(Tnat)]; mtnat<-mean(Tnat)
k10<-length(Tnat);k10
stdmtnat<-var(Tnat)^0.5/k10^0.5
stdmtnat/mtnat*100; message("Mtnat:",mtnat);var(Tnat)
Tcomb<-Tcomb[!is.na(Tcomb)]
k3<-length(Tcomb);k3; mt1<-mean(Tcomb)
std.mt1<-var(Tcomb)^0.5/k3^0.5; std.mt1/mt1*100
message("Mean Tcomb:",mt1);var(Tcomb)
```

Appendix 2 The R-script "2-Parts_VG_2"

```
library(combinat);library(nilde);library(microbenchmark);options(warn=-1)
genSample<-function(M,m,B){
x<-array(0,dim=M*m);R1<-floor(B/(M+1))+1;R2<-floor(B/(M-1))-1
x<-sample(R1:R2,M*m,replace=FALSE)
y<-sum(x); z1<-y==B*m
while(z1!=TRUE){
x<-sample(R1:R2,M*m,replace=FALSE)
y<-sum(x); z1<-y==B*m;};xx<-sort(x,decreasing=TRUE)
d2<-M*m; d4<-M; d6<-B;d1<-list(xx,d2,d4,d6,m)
return(d1);}

k<-5000
Tnat<-rep(NA,k);Tcomb<-rep(NA,k);set.seed(852); Tb<-proc.time()[3]
for(i in 1:k){
ns<-genSample(4,3,500)
d4<-ns[[3]]; d6<-ns[[4]];d7<-ns[[1]];d2<-ns[[2]];d8<-ns[[5]]
g<-get.subsetsum(a=d7,n=d6,M=d4,problem="subsetsum01")
if(g$p.n==d8){
dv2<-rep(0,d2-1);for(i1 in 1:(d2-1)){
dv2[i1]<-d7[1]-d7[i1+1];};dv2<-c(dv2); a4<-d4*d7[1]-d6
T30<-microbenchmark(g<-get.subsetsum(a=dv2,n=a4,M=d4,problem="subsetsum01"),times=1L)
T301<-T30$time/1000000000
Tnat[i]<-T301; b10<-g$solutions
s0<-rep(0,d8);for(j in 1:d8){;s0[j]<-d4-sum(b10[,j]);}
b11<-rbind(s0,b10);row.names(b11)<-paste("s",1:d2,sep="")
sum1<-rep(0,d2);for(k in 1:d2){;sum2k<-sum(b11[k,])
if(sum2k==1){;sum1[k]<-1;};};e1<-sum(sum1);if(e1==d2){
print(d7); print(sum(d7)); print(b11)
a2<-microbenchmark(a4<-combn(x=d7,m=d4,fun=NULL,simplify=TRUE),times=1L)
T1<-a2$time/1000000000; a3<-as.matrix(a4)
a5<-microbenchmark(Solcomb<-as.matrix(a3[,colSums(a3)==d6]),times=1L)
T2<-a5$time/1000000000; print(Solcomb)
Tcomb[i]<-T1+T2;};};}
Te<-proc.time()[3]; print(Te-Tb)
Tnat<-Tnat[!is.na(Tnat)]; mtnat<-mean(Tnat)
k10<-length(Tnat);k10
stdmtnat<-var(Tnat)^0.5/k10^0.5
stdmtnat/mtnat*100; message("Mtnat:",mtnat);var(Tnat)
Tcomb<-Tcomb[!is.na(Tcomb)]
k3<-length(Tcomb);k3; mt1<-mean(Tcomb)
std.mt1<-var(Tcomb)^0.5/k3^0.5; std.mt1/mt1*100
message("Mean Tcomb:",mt1);var(Tcomb)
```

Appendix 3 The R-script "3-Parts_VG_91"

```
library(combinat); library(microbenchmark)
set.seed(988); Tb<-proc.time()[3]
genSample<-function(M,m,B){
x<-array(0,dim=M*m); R1<-floor(B/(M+1))+1; R2<-floor(B/(M-1))-1
x<-sample(R1:R2,M*m,replace=TRUE)
d4<-M; d6<-B; d1<-list(d4,d6,x)
return(d1);}
k<-1000
Tcomb<-rep(NA,k)
for(i in 1:k){
ns<-genSample(4,4,75)
d4<-ns[[1]]; d6<-ns[[2]]; d7<-ns[[3]]
a2<-microbenchmark(a4<-combn(x=d7,m=d4,fun=NULL,simplify=TRUE),times=1L)
T1<-a2$time/1000000000; a3<-as.matrix(a4)
a5<-microbenchmark(Solcomb<-as.matrix(a3[,colSums(a3)==d6]),times=1L)
T2<-a5$time/1000000000;Tcomb[i]<-T1+T2;}
Te<-proc.time()[3];print(Te-Tb)
Tcomb<-Tcomb[!is.na(Tcomb)]
k3<-length(Tcomb);k3
mt1<-mean(Tcomb);mt1
std.mt1<-var(Tcomb)^0.5/k3^0.5
std.mt1/mt1*100
message("Mean Tcomb:",mt1);var(Tcomb)
```