Central Asian Business Journal
Volume 23, No. 1, pp. 12-31
https://doi.org/

An Experimental Supported by Some Strong Theoretical Arguments Proof of the Fundamental Equality P=NP

Vassilly Voinov[1]

Abstract: *A refined version of the algorithm for constructing all 0-1 integer solutions for linear Diophantine equations is introduced. The correctness and its polynomial time complexity is confirmed by a Monte Carlo experiment that uses the trivial simple combinatorics. An implementation of this algorithm proves the polynomial time solvability of many business decision and optimization problems that are currently considered as being intractable: bin packing and cutting stock, open-shop and job-shop scheduling, production planning, dynamic storage allocation and many others. It is shown that all these problems admit polynomial time algorithms for their solution. An experimental proof of the important for the theory fact that every planar symmetric graph possesses at least one Hamiltonian circuit is given.*

A much faster exact polynomial time algorithm for solving symmetric traveling salesman problems is suggested. The algorithm shows that such problems may have several optimal solutions. Numeric randomly generated instances with such solutions that can be used as test examples for newly developed algorithms are presented.

Results of this research present a reliable experimental proof of the fundamental equality P=NP.

AMS subject classifications (2010): *05A18, 05C45, 90C10, 90C27, 11D04, 68Q15, 68Q17*

Keywords: *Linear Diophantine equations, integer linear programming, combinatorial optimization, partitions, polynomial time algorithms, Hamiltonian cycles, traveling salesman problems, P=NP*

1. Introduction

Stephen Cook (2005), p.1 wrote “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”. He noted also that “the importance of the P vs NP question stems from the successful theories of NP-completeness and complexity-based cryptography, as well as the potentially stunning practical consequences of a constructive proof of P=NP”. Important questions related to this problem were mentioned first in the 1930s. During about 95 years

[1] Independent scholar, Kazakhstan. Email: voinovv@mail.ru

mathematicians tried to solve the problem, but even today “the experts and the scientific community does not seem of being able to decide if the P versus NP problem has been solved or not” (Kyritis (2021)).

Woeginger (2016) listed 116 publications devoted to the above problem. 52 works “solve” the P≠NP, 61 “solve” the P=NP, and 3 decides that the problem is unprovable. 4 out of 61 (P=NP) publications that are closely related to this research are: LaPlante (2015), Cardenas et al (2015),Wen-Qi (2012), Zeilberger (2009). It is worth to mention several works published after 2016: Voinov (2017), Voinov et al (2018), Panyukov (2018), Voinov (2019), Voinov and Rahmanov (2020), Müller (2020), Voinov and Pya Arnqvist (2021), Voinov (2022).

The paper is organized as follows: in Section 2 the basic algorithm for solving 0-1 linear Diophantine equation is introduced. The properties of that algorithm are studied in Section 3. Sections 4 and 5 provide a revision and refining of results concerning 2 and 3-partitions obtained in Voinov and Rahmanov (2020). Hamiltonian cycles in symmetric graphs are discussed in Section 6. Section 7 introduces a new approach for solving symmetric traveling salesman problems. A discussion and conclusions are given in Section 8. Appendices provide numerical examples and the software used in this research.

2. Mathematical background

Given arbitrary positive integers $a_1, a_2, \dots, a_n, n \in \mathbb{N}$, consider the problem of representing a positive integer B as a sum of at most $M \le n$ parts such that

$$s_1 a_1 + s_2 a_2 + \cdots + s_n a_n = B, \tag{2.1}$$

where $s_1 + \cdots + s_n \le M$ and $\mathrm{s_i}$ are nonnegative integers. The sign $[x]$ denotes the greatest integer part of x.

Theorem 1. The number of compositions (partitions taking order of summands into account)

$$R_B(M,n) = \sum_{s_n=0}^{\min\{M,\left[\frac{B}{a_n}\right]\}} \sum_{s_{n-1}=0}^{\min\{M,\left[\frac{B-s_n a_n}{a_{n-1}}\right]\}} \cdots \sum_{s_2=0}^{\min\{M,\left[\frac{B-s_n a_n-\cdots-s_3 a_3}{a_2}\right]\}} \frac{M!}{(M-s_1-\cdots-s_n)!s_1!\cdots s_n!} \tag{2.2}$$

if $s_1 + \cdots + s_n \le M, s_1 = (B - s_n a_n - \cdots - s_2 a_2)/a_1$ being a nonnegative integer, and is zero otherwise. The compositions themselves are written down as

$$\{a_1^{s_1}, a_2^{s_2}, \dots, a_n^{s_n}\}, \tag{2.3}$$

wher**e** $\{s_2, \dots, s_n\}$ are sets of summation indices in (2.2). The notation (2.3) means that in any composition there will be s_1 summands of a_1, s_2 of them will be a_2, and so on. Note that $R_B(M,n)$ in (2.2) is a refined expression (11.7) of Voinov and Nikulin (1997), p.156.

Proof. The generating function for the number $R_B(M,n)$ of compositions (partitions taking order of summands into account) of B such as in (2.1) is (Voinov and Nikulin (1995))

$$\Psi_{\boldsymbol{a}}(z) = (1 + z^{a_1} \cdots + z^{a_{n-1}} + z^{a_n})^M = \sum_{l=0}^{M \max\limits_{1\le i\le n}\{a_i\}} R_B(M,n) z^l. \tag{2.4}$$

Writing $\Psi_{\boldsymbol{a}}(z)$ as $[(1 + z^{a_1} \cdots + z^{a_{n-1}}) + z^{a_n}]^M$ and applying the binomial formula, we get

$$\Psi_{\boldsymbol{a}}(z) = \sum_{k=0}^{M} \binom{M}{k} z^{a_n k} (1 + z^{a_1} + \cdots + z^{a_{n-1}})^{M-k} =$$

$$= \sum_{k=0}^{M} \binom{M}{k} z^{a_n k} \sum_{t=0}^{(M-k)\max\limits_{1\le i\le n-1}\{a_i\}} R_t(M-k, n-1) z^t.$$

Changing the order of summation one gets

$$\Psi_{\boldsymbol{a}}(z) = \sum_{l=0}^{M\max\limits_{1\le i\le n}\{a_i\}} \left\{ \sum_{s_n=0}^{\min\{M,\left[\frac{B}{a_n}\right]\}} \binom{M}{s_n} R_{B-s_n a_n}(M-s_n, n-1) \right\} z^l. \qquad (2.5)$$

From (2.4) and (2.5) we obtain the recurrence relation

$$R_B(M,n) = \sum_{s_n=0}^{\min\{M,\left[\frac{B}{a_n}\right]\}} \binom{M}{s_n} R_{B-s_n a_n}(M-s_n, n-1). \qquad (2.6)$$

This relation gives

$$R_B(M,n) = \sum_{s_n=0}^{\min\{M,\left[\frac{B}{a_n}\right]\}} \sum_{s_{n-1}=0}^{\min\{M,\left[\frac{B-s_n a_n}{a_{n-1}}\right]\}} \cdots \sum_{s_2=0}^{\min\{M,\left[\frac{B-s_n a_n-\cdots-s_3 a_3}{a_2}\right]\}} \binom{M}{s_n}\binom{M-s_n}{s_{n-1}}\cdots$$

$$\cdots\binom{M-s_n-\cdots-s_3}{s_2} R_{B-s_n a_n-\cdots-s_2 a_2}(M-s_n-\cdots-s_2, 1).$$

It can be shown (Voinov and Nikulin (1997), p.155) that $R_{B-s_n a_n-\cdots-s_2 a_2}(M-s_n-\cdots-s_2,1) = \binom{M-s_n-\cdots-s_2}{s_1}$ if $s_1 = (B - s_n a_n - \cdots - s_2 a_2)/a_1$ is a nonnegative integer, and is zero otherwise. Since

$$\binom{M}{s_n}\binom{M-s_n}{s_{n-1}}\cdots\binom{M-s_n-\cdots-s_2}{s_1} = \frac{M!}{(M-s_n-\cdots-s_1)!s_1!\cdots s_n!}, \qquad (2.7)$$

we finally obtain (2.2).

Since terms (2.7) count all compositions of B for fixed sets $\{s_1,\dots,s_n\}$, the compositions themselves are enumerated as in (2.3).

Corollary 1. Equating terms (2.7) in (2.2) to 1 we obtain the number of partitions for B on at most $M \le n$ parts as

$$R_B(M,n) = \sum_{s_n=0}^{\min\{M,\left[\frac{B}{a_n}\right]\}} \sum_{s_{n-1}=0}^{\min\{M,\left[\frac{B-s_n a_n}{a_{n-1}}\right]\}} \cdots \sum_{s_2=0}^{\min\{M,\left[\frac{B-s_n a_n-\cdots-s_3 a_3}{a_2}\right]\}} 1, \qquad (2.8)$$

where $s_1 + \cdots + s_n \le M$ and $s_1 = (B - s_n a_n - \cdots - s_2 a_2)/a_1$ being a nonnegative integer. As previously, the partitions (solutions of the equation (2.1)) are defined by (2.3).

Corollary 2. Consider the subset sum problem. The solution of this problem for any sum $B \in \mathbb{Z}^+$ means to find a vector $(s_1,\dots,s_n)^T$ with $s_i \in \{0,1\}$, such that

$$s_1 a_1 + s_2 a_2 + \cdots + s_n a_n = B. \qquad (2.9)$$

The number of the equation (2.9) solutions on at most $M \le n$ parts equals

$$R_B(M,n) = \sum_{s_n=0}^{\min(1,\left[\frac{B}{a_n}\right])} \sum_{s_{n-1}=0}^{\min(1,\left[\frac{B-s_n a_n}{a_{n-1}}\right])} \cdots \sum_{s_2=0}^{\min(1,\left[\frac{B-s_n a_n-\cdots-a_3 s_3}{a_2}\right])} 1 \qquad (2.10)$$

for $s_1 + \cdots + s_n \leq M$ and $s_1 = (B - s_n a_n - \cdots - s_2 a_2)/a_1$ being 0 or 1, and is zero otherwise. All solutions themselves, if exist, can be enumerated as $\{a_1^{s_1}, \dots, a_n^{s_n}\}$, where $\{s_2, \dots, s_n\}$ are sets of summation indices in (2.10). If $M = n$ then the algorithm in (2.10) will enumerate all solutions of the equation (2.9).

Being based on a generating function for partitioning natural numbers on parts with positive integer components the algorithm in (2.10) will be called as GFA.

3. A note on solutions of a 0-1 linear Diophantine equation

Consider first the following simplest intuitive combinatorial algorithm (ICA) for enumerating all 0-1 solutions of the equation in (2.9). To do this we have to get all 2^n combinations of a vector $(a_1, \dots, a_n)^T$, and select those of them that satisfy the equation (2.9). This can be done, e.g., by using the lexicographic or equivalent to it the revolving door algorithm of Nijenhuis and Wilf (1978) realized in the R-package "combinat".

As an alternative one may use the GFA for $M = n$. This algorithm uses the R-function "get.subsetsum" from the R-package "nilde". If in (2.10) there were $n-1$ sums from 0 to 1, then the expected time complexity of that algorithm would be $O(2^{n-1})$. But, actually, some upper limits of sums (dependent on $a_1, a_2, \dots, a_n$) can be zero, and the total number of those simple operations will be less than 2^{n-1}. If instances under consideration are random, then it will be not easy to assess that number. In view of this the R-command "microbenchmark" was used to define the computing time of algorithms used. This software computes the CPU time with the accuracy of one nanosecond.

To assess the complexity of both approaches numerically consider the following computer experiment. For all 15 pairs $[n, B]$ with $n \in [6,9,\dots,48]$ and $B \in [45,60,\dots,255]$ ([6,45], [9,60],…, [48,255]) random samples were generated using the R-command "sample(1:*B*,*n*,replace=FALSE)". To generate a sample of size n this command uses different positive integers uniformly distributed in the range $[1, B]$. An application of "microbenchmark" permits to estimate the computing time for solutions constructed by the ICA and the GFA with a small relative standard deviation of the mean δ. For 4,000 samples generated by a standard PC (Intel® Core(™) i7-2600 CPU@3.40 GHz, RAM 24.00GB) the algorithms give $\delta < 2.5\%$. Results of the simulation are given in Table 1. Note that the lack of RAM did not permit to use ICA for $n > 30$.

Table 1. Mt1 - mean time in seconds needed to enumerate all solutions of the equation in (2.9) using the GFA. Mt2 – same for the ICA.

n	*p=n+B+nB*	Mt1	Mt2		*n*	*p=n+B+nB*	Mt1	Mt2
6	321	0.00080454	0.00074977		30	5145	0.1009730	1490.8397
9	609	0.00160270	0.00165080		33	6153	0.1646100	-
12	987	0.00316990	0.00649209		36	7251	0.2675440	-
15	1455	0.00601060	0.04359170		39	8439	0.4154940	-
18	2013	0.01121000	0.34672344		42	9717	0.6412170	-
21	2661	0.02065830	2.82593890		45	11085	0.9521615	-
24	3399	0.03622090	22.7577757		48	12543	1.4013820	-
27	4227	0.06013500	182.794950					

For every $n \in [6,9,\dots,48]$ and $B \in [45,60,\dots,255]$ the ICA and the GFA produce **the same solutions** for all generated instances. These results confirm the correctness of the algorithm GFA. An interested reader may reproduce the above results using his/her own seed and the R-script "GFA and combn". This script can be requested for free by the email voinovv@mail.ru.

From Table 1 one sees that the GFA is much faster than the ICA. For illustration consider, e.g., the randomly generated vector $A=(94,73,128,17,24,133,63,64,86,72,151,36,104,83,135,99,51,58,75,59,127,77,115,142,85,82,25,9,89,162)^T$ of size $n=30$. For $B=165$ both algorithms produce the same 24 solutions: {83,82}, {73,17,75}, {73,83,9}, {17,63,85}, {17,59,89}, {24,64,77}, {24,83,58}, {24,59,82}, {63,77,25}, {36,104,25}, {51,25,89}, {58,82,25}, {73,17,24,51}, {73,24,59,9}, {73,58,25,9}, {17,24,99,25}, {17,24,115,9}, {17,64,75,9}, {17,64,59,25}, {17,72,51,25}, {72,59,25,9}, {17,24,63,36,25}, {17,24,64,51,9}, and {17,63,51,25,9}. The computing time for the ICA is 1,495.22 sec., and 0.03285 sec. for the GFA.

Garey and Johnson (1978), p. 500 noted that "… for most formal complexity theory, the time complexity of an algorithm is expressed in terms of a single "instance size" parameter which reflects the number of symbols that would be required to describe the instance in a "reasonable" and "concise" manner". An instance in (2.9) is well described by the parameter $p=n+B+nB$. Note that nB is related to the upper bound for the number of symbols required to describe the left-hand side part of the equation (2.9).

The simulated mean times' curves fitted by Microsoft Excel 2018 for Mt1 and Mt2 are presented on Figures 1 and 2 respectively.

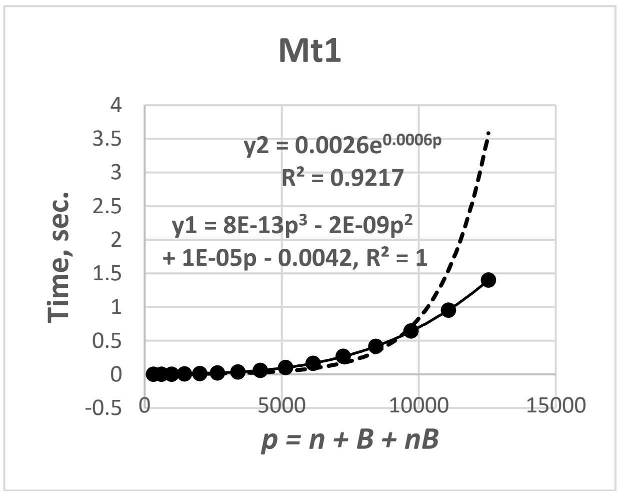


Fig. 1

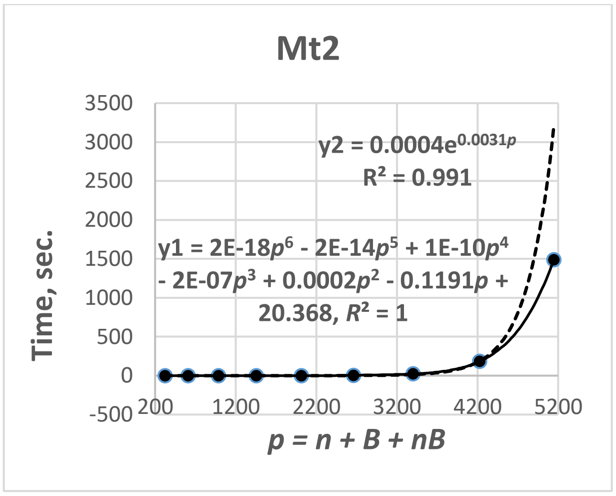


Fig. 2

From Fig. 1 we see that the degree 3 polynomial $y1 = 8E\text{-}13p^3 - 2E\text{-}09p^2 + 1E\text{-}05p - 0.0042$ (solid line with multiple $R^2=1$) provides an excellent fit for the CPU time dependence on p. It evidently declines the exponential fit $y2 = 0.0026e^{0.0006p}$ (dashed line with $R^2=0.9217$). Two other criteria (Residual Standard Error (RSE) and Akaike's Information Criterion (AIC)) confirm this conclusion. RSE and AIC for y1 equal 0.002164 (on 11 degrees of freedom (d.f.)) and -136.1568 respectively, but for y2 they are 0.008934 and -95.1157 correspondingly. From the above follows $O(p^3)$ polynomial time solvability of a 0-1 linear Diophantine equation (2.9). Thus, Zero-One Integer Programming is not only NP-complete as it was stated by Garey and Johnson (1979), p. 245, but belongs to class P as well.

Fig. 2 represents results of fitting CPU times obtained by ICA. One sees that formally the polynomial fit $y1 = 2E\text{-}18p^6 - 2E\text{-}14p^5 + 1E\text{-}10p^4 - 2E\text{-}07p^3 + 0.0002p^2 - 0.1191p + 20.368$ with $R^2=1, \mathrm{RSE}=1.907, \mathrm{AIC}=39.62$ is better than the exponential one $y2 = 0.0004e^{0.0031p}$ with $R^2=$

$0.991, \mathrm{RSE} = 1.945, \mathrm{AIC} = 41.25$. Since there is no essential difference in criteria values, the above conclusion is not reliable. It can be explained, e.g., by the fact that Fig. 2 is based on only 9 experimental values versus 15 for Fig. 1.

Remark 3.1. The results of this simulation experiment can be summarized as follows:
a) Both the GFA and the ICA produce all absolutely the same 0-1 solutions of the equation (2.9). This important result confirms the correctness of the GFA.
b) The complexity of the GFA is estimated as $O(p^3)$, where instances' "size" $p = n + B + nB$. The GFA is much faster as compared to ICA. If, e.g., $n = 30$ and $B = 165$, then the computing time for the GFA is about 15,000 times less than that for the ICA.
c) The 0-1 Integer Linear Programming (ILP) problem belongs to class P.
d) Since there is a polynomial time reduction from the 3-Satisfiability (3SAT) to ILP problem (see Karp (1972), p.97 and Garey and Johnson (1979), p.245), then, due to Lemma 2.1 of ibid, pp.34-35, the **3SAT problem belongs to the class P** (see also Remark 6.3). A constructive algorithm of Müller (2020) confirms this conclusion.

4. 2-Partition revisited

Voinov and Rahmanov (2020) introduced and considered the following formulation of the 2-Partition problem:
Instance: A finite set A of $2m$ elements, a bound $B \in \mathbb{Z}^+$, and a "size" $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that each $s(a)$ satisfies $B/3 < s(a) < B$, and such that $\sum_{a \in A} s(a) = mB$.
Question: Can A be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ such that for $1 \leq i \leq m$ $\sum_{a \in S_i} s(a) = B$. (The above constraints on the item sizes imply that every such S_i will contain exactly two elements from A).

This formulation is analogous to that of 3 and 4-Partition problems formulated by Garey and Johnson (1979, pp. 96-97. In Voinov and Rahmanov (2020) the authors considered the restricted 2-Partition problem. Using the R-function "nlde" from the R-package "nilde" they have provided an experimental proof of the fact that if B is fixed, then the time complexity is estimated as $O(n^6)$. Thus, actually, their algorithm was pseudo-polynomial. It has to be noted that the R-function "nlde" is less effective than the function "get.subsetsum" used in section 3. Because of this it is desirable to revisit an assessment of the time complexity of algorithms for solving the 2-Partition problem. To do this consider the following experiment based on the GFA that uses the function "get.subsetsum", and ICA that uses the function "combn". Since both algorithms use solutions of equations (2.9), then, as in section 3, in the following experiments the parameter $p = n + B + nB$ will be used.
Exp.1. For every 9 pairs $[n, B]$ with $n \in [4,6, \ldots, 20]$ and $B \in [40,50, \ldots, 120]$ 4,000 samples were generated at random using the R-command "sample(*R1:R2*,2*m*,replace=FALSE)", where $R1 = [B/3] + 1$ and $R2 = B - 1$. This command uses different positive integers from the range $[R1, R2]$ to generate a random sample of size $n = 2m, m = 2,3, \ldots, 10$. Times needed to enumerate solutions of the equation (2.9) on exactly 2 parts were estimated by R-command "microbenchmark()" that estimates the computing time for solutions containing exactly 2 parts constructed by the GFA and that by the ICA with a small relative standard deviation of the mean δ. For 4,000 samples δ was less than 2.3%. Results of the simulations are provided in Table 2 and Figures 3 and 4 below.

Table 2. Mt3 – mean time in seconds needed to enumerate all solutions of the equation in (2.9) using the GFA. Mt4 – same for the ICA.

n	*B*	*p=n+B+nB*	Mt3	Mt4
4	40	204	0.00052816	0.00011587
6	50	356	0.00079348	0.00012211
8	60	548	0.00126969	0.00013219
10	70	780	0.00192497	0.00014253
12	80	1052	0.00288650	0.00015709
14	90	1364	0.00420054	0.00017263
16	100	1716	0.00601045	0.00019322
18	110	2108	0.00818904	0.00021668
20	120	2540	0.01083942	0.00024617

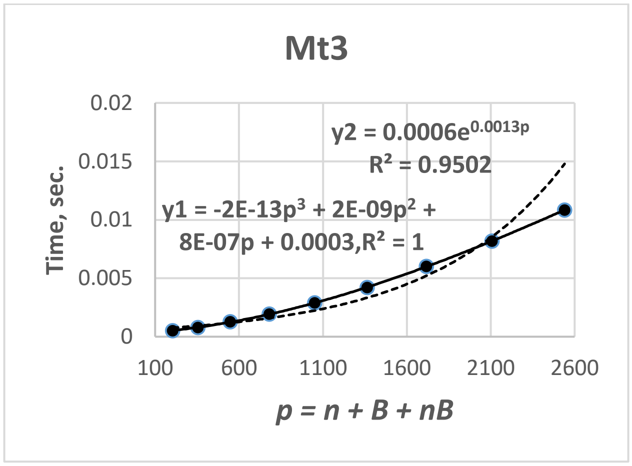


Fig. 3

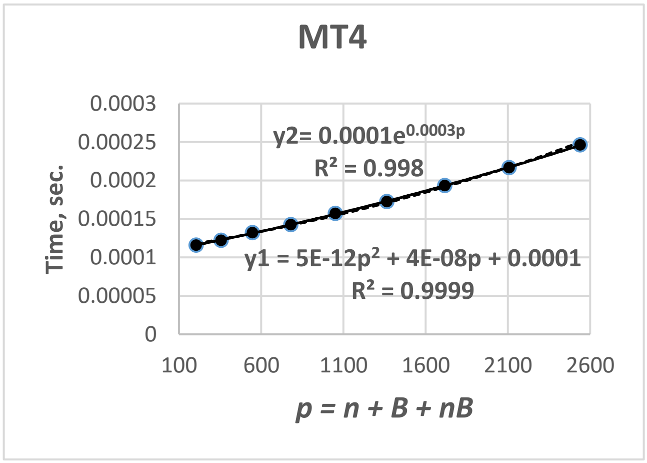


Fig. 4

From these figures we see that the multiple R^2 of polynomial fits (solid lines) is one or almost one for both approaches. At the same time for the exponential fits (dashed lines) the values of R^2 are essentially less than one. Thus we have to conclude that both GFA and ICA are polynomial in time. Criteria RSE and AIC for Mt3 confirm this. RSE and AIC for y1 = -2E-13p^3 + 2E-09p^2 + 8E-07p + 0.0003 equal $2.3 \cdot 10^{-5}$ (on 5 d.f.) and -161.8932 respectively, but for $y2 = 0.0006e^{0.0013p}$ they are 0.0006232 and -103.5734 correspondingly. The same situation is for Mt4. RSE and AIC for $y1 = 5E - 12p^2 + 4E - 08p + 0.0001$ equal $6.221 \cdot 10^{-7}$ and -227.3311 respectively, but for $y2 = 0.0001e^{0.0003p}$ they are $1.964 \cdot 10^{-6}$ (on 6 d.f.) and -207.2489 correspondingly. These results provide the conclusion that 2-Partition problem is solvable in polynomial time with the time complexity of $O(p^{\leq 3})$.

Remark 4.1. Since the above solution of the 2-Partition problem is based on consisting of exactly two parts solutions of a 0-1 linear Diophantine equation considered in Section 3, both the GFA and the ICA realized in the R-script "two_parts" of Appendix 1 produce the same partitions. If, for example, $m = 3, B = 50, n = 6$, then the script gives 125 answers "yes" out of 4,000 random instances. Two of them are given below for illustration.

```
[1] 18 20 21 29 30 32            [1] 17 19 22 28 31 33        # random samples (A)
[1] 150                          [1] 150                      # ∑_{a∈A} s(a) = mB
   sol.1 sol.2 sol.3                sol.1 sol.2 sol.3         # solutions by GFA
s1    0     0     1              s1    0     0     1
```

```
s2    0   1   0                  s2    0   1   0
s3    1   0   0                  s3    1   0   0
s4    1   0   0                  s4    1   0   0
s5    0   1   0                  s5    0   1   0
s6    0   0   1                  s6    0   0   1
     [,1] [,2] [,3]                   [,1] [,2] [,3]          # solutions by ICA
[1,]  18   20   21               [1,]  17   19   22
[2,]  32   30   29               [2,]  33   31   28
```

Remark 4.2. The 2-Partition problem is a particular case of the classical Partition problem defined by Garey and Johnson (1979), p.223 as follows:
Instance: Finite set A and a size $s(a) \in \mathbb{Z}^+$ for every $a \in A$.
Question: Is there a subset $A' \subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a)$?
Under such definition of Partition parts A' and $A - A'$ may have both the same number of elements in each part (e.g., two for 2-Partition above) or different number of them. Consider the situation typical to the numerical example of (ibid, p.91). They considered the following artificial “instance of Partition for which $A = \{a_1, a_2, a_3, a_4, a_5\}, s(a_1) = 1, s(a_2) = 9, \; s(a_3) = 5, s(a_4) = 3,$ and $s(a_5) = 8$". Note that $\sum_{a\in A} s(a) = B = 26$. Using the intuitive “dynamic programming” simple procedure with the complexity $O(nB)$ they have shown that the partition with $A' = \{1,9,3\}$ and $A - A' = \{5,8\}$ satisfies the condition $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a) = 13$, thus, answering “yes” on the **Question**. ”Since “nB” is not bounded by any polynomial function of this quantity”, Garey and Johnson concluded that Partition belongs to class NP. This wrong conclusion is disproved by the following experiment based on ICA.
Let, as above, $\sum_{a\in A} s(a) = B = 26$ with $s(a) \in \{1,2,\ldots,10\}$. Constructing, e.g., 50 random samples A of size 5 and using the R-script “two_parts” one will get 39 partitions that answer “yes”. Two of them are given below for illustration.

```
[1] 1 3 6 7 9          [1] 3 4 5 6 8         # random samples (A)
[1] 26                 [1] 26                # ∑_{a∈A} s(a) = B
  sol.1 sol.2            sol.1 sol.2         # solutions by GFA
s1    0    1           s1    1    0
s2    0    1           s2    1    0
s3    1    0           s3    0    1
s4    1    0           s4    1    0
s5    0    1           s5    0    1
     [,1] [,2]              [,1] [,2]        # solutions by ICA
[1,]    6    1         [1,]    5    3
[2,]    7    3         [2,]    8    4
[3,]         9         [3,]         6
```

Note that the artificial partition ({1,9,3},{5,8}) of Garey and Johnson also presents in that set of 39 random instances. Since the complexity of ICA is given by the degree two polynomial of nB, one concludes that the Partition problem is in class P. This fundamental result is very important for both the theory and applications to practical problems because the complexity of many applied problems is explained by the polynomial-time transformation from Partition. This means that if such a transformation exists, then (even if that problem is currently considered to be NP-hard) a polynomial-time algorithm solving the problem can be developed. Based on “an extensive list of known NP-complete and NP-hard problems” of (ibid, Ch. 7) a short list of problems polynomially reducible from

Partition is given below (references to corresponding pages of Chapter 7 are presented in parentheses): a) Subset sum (p. 223), b) Bin packing (p. 226), c) Sequencing to minimize tardy task weight (p. 236), d) Sequencing with deadlines and set-up times (p. 238), e) Open-shop scheduling (p. 241), f) Production planning (p. 243), g) Quadratic programming (p. 245), h) Knapsack (p. 247).

Remark 4.3. The paper of Zeilberger (2009) confirms the above conclusion for the Subset sum problem. The author developed an original computer-generated proof based on polynomial time algorithm. He solved "more than 10,000 linear programming problems, each with more than 100,000 variables" and obtained the answer "yes" on the problem's Question. From this the author deduced that P=NP.

5. 3-Partition revisited

The 3-Partition problem is defined as follows (Garey and Johnson (1979), p.96):
Instance: A finite set $A = \{a_1, a_2, \dots, a_{3m}\}$ of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a "size" $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that each $s(a)$ satisfies $B/4 < s(a) < B/2$, and such that $\sum_{a\in A} s(a) = mB$.
Question: Can A be partitioned into m disjoint sets $S_1, S_2, \dots, S_m$ such that for $1 \le i \le m, \sum_{a\in S_i} s(a) = B$. (Notice that the above constraints on the item sizes imply that every such S_i must contain exactly three elements from A).

Voinov and Rahmanov (2020) showed that the answer on this **Question** is "yes", on the contrary of the commonly accepted "no". Using the algorithm that is similar to but less effective than GFA they have simulated numerous counterexamples.

Exp.2. For two pairs of $[m, B]$ with $m = 2{,}3$ and $B = 90$ samples $A = \{a_1, a_2, \dots, a_{3m}\}$ were randomly generated using the R-command "sample(*R1:R*2,3*m*,replace=TRUE)", where $R1 = [B/4] + 1 = 23, R2 = [B/2] - 1 = 44,$ and $\sum_{i=1}^{3m} a_i = mB, m = 2{,}3, \dots$ This command generates random samples of size $3m$ using positive integers from the range $[R1, R2]$. Note that some numerical values of a_i may be used several times, and that all conditions of an **Instance** are satisfied.

For any instance the solutions of the equation (2.9) for $M = 3$ were obtained using the GFA and the ICA.

The simulation results are as follows:
1) As in Sections 3 and 4 both approaches produce the same 0-1 solutions for all generated samples,
2) If $m = 3$, then 4,327 random instances out of 46,573 ones answer "yes",
3) If $m = 4$, then 49 random instances out of 5,601 ones answer "yes".

A particular generated at random numerical example that illustrates the above results is:
Let $m = 3$, $B = 90$ and $A = \{37,23,24,31,29,27,32,25,42\}$. The following 3 disjoint subsets answering "yes" on the **Question** are: {37,24,29}, {23,25,42}, {31,27,32}.

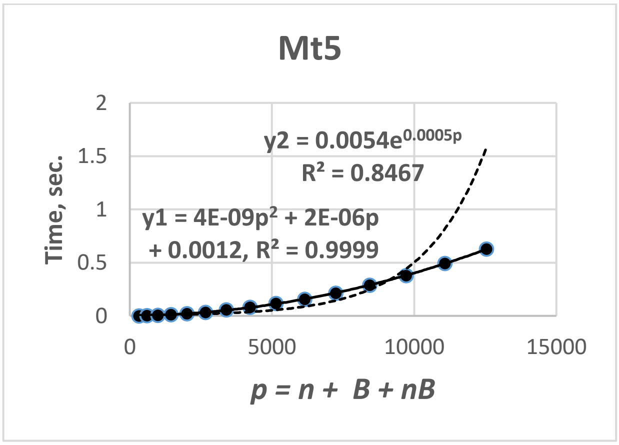
Due to the definition in (ibid, p. 18) "a decision problem Π consists of a set D_Π of instances and a subset $Y_\Pi \in D_\Pi$ of yes-instances" the results of the Exp.2 solve the 3-Partition decision problem.

To assess the complexity of algorithms used consider the next experiment.

Exp.3. For all 16 pairs of $[n, B]$ with $n \in [6, 9, \dots, 48]$ and $B \in [45, 60, \dots, 255]$ 1,000 samples were generated using the R-command "sample(*R1:R2*,3*m*,replace=FALSE)". This command uses different positive integers from the range $[R1, R2]$ to generate a random sample of size $n = 3m, m = 2, 3, \dots, 16$. Times needed to enumerate solutions of the equation (2.9) on exactly 3 parts were estimated by R-command "microbenchmark()". An application of this software permits to estimate the computing time for solutions containing exactly 3 parts constructed by the GFA and by the ICA with a rather small relative standard deviation of the mean δ. For 1,000 samples δ was less than 0.8%. Numerical results of the simulations are provided in Table 3 and Figures 5 and 6 below.

Table 3. Mt5 - mean time in seconds needed to enumerate solutions on exactly 3 parts of the equation in (2.9) using the GFA. Mt6 - same for the ICA.

n	*B*	*p=n+B+nB*	Mt5	Mt6
6	45	321	0.00107457	0.000140646
9	60	609	0.00271464	0.000194590
12	75	987	0.00617127	0.000290530
15	90	1455	0.01199900	0.000472490
18	105	2013	0.02104006	0.000726200
21	120	2661	0.03401750	0.001075750
24	135	3399	0.05630800	0.001565000
27	150	4227	0.08096600	0.002179000
30	165	5145	0.11610500	0.002971000
33	180	6153	0.15631400	0.003895600
36	195	7251	0.21235000	0.005031600
39	210	8439	0.28815000	0.006352000
42	225	9717	0.37803400	0.007659000
45	240	11085	0.49332000	0.009298000
48	255	12543	0.62674580	0.011124000



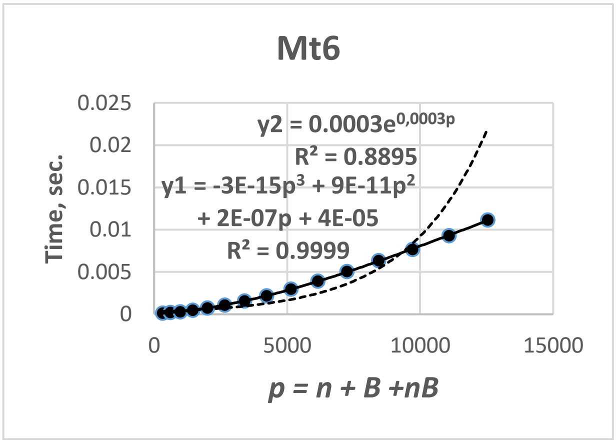


Fig. 5 Fig. 6

From these figures we see that the multiple R^2 of polynomial fits (solid lines) is almost one for both approaches. At the same time for the exponential fits (dashed lines) the values of $R^2$are essentially less than one. Thus we have to conclude that both GFA and ICA are polynomial in time. Criteria RSE and AIC for Mt5 confirm this. RSE and AIC for y1 = 4E-09p^2 + 2E-06p + 0.0012 equal 0.00245 (on 12 d.f.) and -133.1254 respectively, but for $y2 = 0.0054e^{0.0005p}$ they are 0.03183 and -57.00329 correspondingly. The same situation is for Mt6. RSE and AIC for $y1 = -3E - 15p^3 + 9E - 11p^2 + 2E - 07p + 4E - 05$ equal $4.35 \cdot 10^{-5}$ (on 11 d.f.) and -253.3688 respectively, but for $y2 = 0.0003e^{0.0003p}$ they are $7.836 \cdot 10^{-4}$ and -168.1272 correspondingly. These results provide the conclusion that 3-Partition problem is solvable in polynomial time with the time complexity of $O(p^{\leq 3})$.

Remark 5.1. Like in case of the Partition this result is very important for both the theory and applications because the complexity of many applied problems is explained by the polynomial-time transformation from the 3-Partition. This means that if such a transformation exists, then (even if that problem is currently considered to be NP-hard) a polynomial-time algorithm solving those problems can be developed.

A list of problems polynomially reducible from the 3-Partition (references to corresponding pages of (ibid, Chapter 7) are presented in parentheses) is:
a) Bandwidth (p. 200), b) Weighted diameter (p. 205), c) Intersection graph for segments on a grid (p. 219), d) Edge embedding on a grid (p. 219), e) Minimum sum of squares (p. 225), f) Bin packing (p. 226), g) Dynamic storage allocation (p. 226), h) Expected retrieval cost (p. 227), i) Sequencing with release times and deadlines (p. 236), j) Sequencing to minimize weighted tardiness (p. 237), k) Resource constrained scheduling (p. 239), l) Open-shop scheduling (p. 241), m) Job-shop scheduling (p. 242).

Remark 5.2. The numerical 3-dimensional matching (N3DM) decision problem is formulated as follows (ibid, p. 224):
Instance: Disjoint sets $W, X,$ and Y, each containing m elements, a size $s(a) \in \mathbb{Z}^+$ for each element $a \in W \cup X \cup Y$, and a bound $B \in \mathbb{Z}^+$.
Question: Can $W \cup X \cup Y$ be partitioned into m disjoint sets $A_1, A_2, \dots, A_m$ such that each A_i contains exactly one element from each of $W, X,$ and Y and such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$?

It is easily seen that this problem is a particular case of 3-Partition problem. The following numerical example clearly explains the situation. Let one apply the approach of **Exp.2** to the randomly generated instance: $m = 3$, $B = 90$ and $A = \{25,26,23,36,40,34,33,29,24\}$. Using $A = W \cup X \cup Y$, where $W = \{25,26,23\}, X = \{36,40,34\}$ and $Y = \{33,29,24\}$ and solving the 3-Partition problem one gets the following three disjoint sets: $A_1 = \{25,36,29\}$, $A_2 = \{26,40,24\}$ and $A_3 = \{23,34,33\}$ each sum to 90. Every A_i contains exactly one element from $W, X,$ and Y thus giving a solution for the decision problem. An experiment similar to **Exp.2** produces as many instances answering "yes" as one needs. The polynomial time solvability of the N3DM problem follows from that for the 3-Partition problem.

The results of this section can be summarized as follows:
1) The GFA and the ICA produce absolutely the same 0-1 solutions of the equation (2.9) on exactly 3 parts.
2) The decision 3-Partition problem has been solved. The simulations produce numerous counterexamples showing that the answer on the problem's **Question** is "yes".
3) The ICA is the most efficient. If, e.g., $\mathrm{n} = 48$, then computing time by ICA is 56.3 times less than that by the GFA.
4) The 3-Partition problem's complexity is $O(p^{\leq 3})$. This means that it is solvable in polynomial time by deterministic algorithms, and, hence, belongs to class P.
5) The numerical 3-dimentional matching problem belongs to class P.

Remark 5.3. The 3-dimensional matching (3DM) problem is formulated as (ibid, p.221):
Instance: Set $M \subseteq W \times X \times Y$, where $W, X,$ and Y are disjoint sets having the same number q of elements.
Question: Does M contain a matching, i.e., a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate?

It is known (ibid, p. 99) that there is a polynomial time transformation from 3DM to N3DM. Since N3DM is solvable in polynomial time, and, hence belongs to class P, from Cook (1971), p. 156 it follows that 3DM also belongs to class P. Since there is a polynomial time reduction from the N3DM to 3-Partition (ibid p. 90), it again follows that 3DM is in P.

6. Symmetric Hamiltonian graphs

The graph $G = (V, E)$ is called Hamiltonian if it has at least one Hamiltonian cycle or circuit. The Hamiltonian cycle (HC) problem is formulated as (Garey and Johnson (1979)):
Instance: A graph $G = (V, E)$.
Question: Does G contain a Hamiltonian circuit, that is, an ordering $< v_1, v_2, \dots, v_n >$ of vertices of G, where $n = |V|$, such that $\{v_n, v_1\} \in E$ and $\{v_i, v_{i+1}\} \in E$ for all $i, 1 \le i < n$?

Without loss of generality a graph can be described by a diagonal distance/cost $n \times n$ matrix $C = (c_{uv})$, where $n(n-1)$ edges $u, v \in E$ are nonnegative integers. Laporte (1972), p. 237 noted that "symmetrical problems are better handled by specialized algorithms that exploit their structure". In view of this the following variant of GFA for solving a symmetric HC problem is proposed:
Alg.1.
Step 1. (Initialization) Apply a heuristic algorithm (e.g., "cheapest_insertion") for the matrix C to get an upper bound Ub for the Hamilton cycle length.
Step 2. Construct all solutions on exactly n parts for the equation (2.9) with $B = Ub$ and $n(n-1)/2$ coefficients that are entries of, say, lower triangular part of C.
Step 3. Select solutions that satisfy conditions of the problem: all vertices should be of degree two, there should be no subtours, and $\{v_n, v_1\}$ must be in E.

A heuristic "cheapest_insertion" algorithm provides a solution of the HC decision problem, but it does not provide the uniqueness and polynomial time solvability of that solution. To investigate properties of the Alg.1 for solving a symmetric HC problem consider the following experiment:
Exp.4. For all 9 pairs of $[n, B]$ with $n \in [3,4, \dots, 11]$ and $B \in [20,30, \dots, 100]$ 4,000 random instances were generated using the R-command "sample(1:*B*,*n***n*,replace=TRUE)". Times needed to perform the above algorithm was estimated by the R-command "microbenchmark()".

Numerical results of the simulations are provided in Table 4 and Figure 7 below. Note that "instance sizes" were described by the parameter $p = n + \frac{B(n^2-n)}{2} + nB = n + Bn(n+1)/2$, which is an upper bound for the actual "instance size".

Table 4. Mtham - mean time in seconds needed to enumerate solutions for given *n* and *B* using the Alg.1.

n	*B*	*p=n+Bn(n+1)/2*	Mtham
3	20	123	0.00252540
4	30	304	0.00341131
5	40	605	0.00980120
6	50	1056	0.07678800
7	60	1687	1.08727400
8	70	2528	13.4450900
9	80	3609	159.064400
10	90	4960	2146.79800
11	100	6611	18683.9830

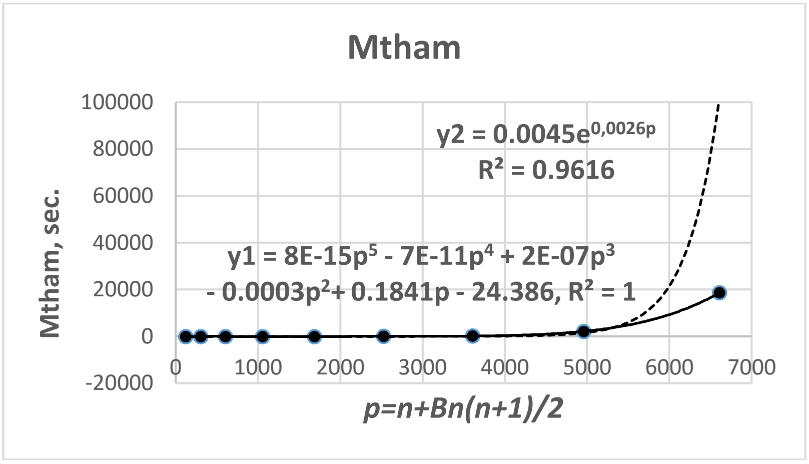


Fig. 7

From this figure we see that the multiple R^2 of the polynomial fit $y1$ (solid line) is one. At the same time for the exponential fit (dashed line) the value of $R^2$is significantly less than one. Thus, one may to conclude that the Alg.1 solves the HC problem in polynomial time with the complexity $O(p^5)$. Criteria RSE and AIC for Mtham confirm this. RSE and AIC for $y1 = 8E - 15p^5 - 7E - 11p^4 + 2E - 07p^3 - 0.0003p^2 + 0.1841p - 24.386$ equal 10.72 (on 3 d.f.) and 72.346 respectively, but for $y2 = 0.0045e^{0.0026p}$ they are 79.82 (on 7 d.f.) and 108.115 correspondingly.

Remark 6.1. All 4,000 random instances of Exp.4 possess at least one Hamiltonian cycle. Many of them have several citcuits of the same length. The random symmetric graph with 9 vertices described by the cost/distance matrix d in Figure 8 illustrates the situation.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]
[1,]	■	50	38	55	45	42	35	60	52
[2,]	50	■	33	21	12	28	22	40	59
[3,]	38	33	■	41	52	47	15	40	41
[4,]	55	21	41	■	16	55	54	29	27
[5,]	45	12	52	16	■	61	11	16	72
[6,]	42	28	47	55	61	■	32	32	28
[7,]	35	22	15	54	11	32	■	27	68
[8,]	60	40	40	29	16	32	27	■	47
[9,]	52	59	41	27	72	28	68	47	■

Fig. 8

The Alg.1 produces 4 different Hamiltonian cycles of length 238: [1] 3 1 6 9 4 2 8 5 7 3, [2] 2 1 3 7 5 8 6 9 4 2, [3] **3 1 9 6 2 4 8 5** 7 3, and [4] 3 1 9 6 8 4 2 5 7 3. Note that the upper bound for Alg.1 given by the R-command "solve_TSP(d,method="cheapest_insertion")" is 238 with a tour **5 8 4 2 6 9 1 3 7 5.** This Hamiltonian cycle is identical to the circuit No. 3 given by the Alg.1. From this it follows that a heuristic "cheapest_insertion" algorithm does not enumerate all existing Hamiltonian cycles. By the way, it does not produce also the optimal tour for this particular symmetric TSP, which, as one will see in the next section, is **3 1 6 9 4 2 5 8 7 3** of length 226.

The **Exp.4** showed that all 4,000 generated at random instances possess at least one Hamiltonian cycle. The same is true for disconnected graphs. These results can be considered as an experimental proof of the following

Conjecture: *Every planar symmetric graph possesses at least one Hamiltonian cycle.*

The same result for asymmetric TSP was observed but not mentioned in Voinov and Pya Arnqvist (2021).

Remark 6.2. Duan Wen-Qi (2012) presented an alternative "constructive algorithm to solve the undirected Hamiltonian cycle problem in polynomial time".
Remark 6.3. The vertex cover (VC) and the Clique problem are defind by Garey and Johnson (1979), pp.46-47 as follows:
Instance: A graph $G=(V,E)$ and a positive integer $K \leq |V|$.
Question: Is there a ***vertex cover*** of size K or less for G, that is, a subset $V' \subseteq V$ such that $|V'| \leq K$ and, for each edge $\{u,v\} \in E$, at least one of u and v belongs to V'?

Instance: A graph $G=(V,E)$ and a positive integer $J \leq |V|$.
Question: Does G contain a ***clique*** of size J or more, that is, a subset $V' \subseteq V$ such that $|V'| \geq J$ and every two vertices in $V'$are joined by an edge in E?
Evidently that every Hamiltonian cycle is both a vertex cover of size $K=|V'|$ and a clique of size $J=|V'|$. Since Alg.1 can be easily adapted for constructing VCs of size $k \leq K$ and Cliques of size $\leq J$, then it follows that the Alg.1 solves the VC and the Clique problem in polynomial time. Hence, both VC and the Clique problems belong to the class P.
Since there is a polynomial time transformation from 3-satisfiability (3SAT) problem to the VC (see ibid, p.190), then due to theorem 4 of Cook (1971), p. 156 we immediately conclude the 3SAT problem is also solvable in polynomial time. This result is confirmed by Matthias Müeller (2020) who has developed an exact polynomial time $O(n^{15})$ algorithm that solves the 3SAT problem.

7. Symmetric traveling salesman problem

The traveling salesman problem (TSP) is usually formulated as: given a graph $G=(V,A)$ with $n=|V|$ vertices and $n(n-1)=|A|$ arcs or edges, and a distance/cost matrix $C=(c_{ij})$ associated with A find minimum Hamiltonian cycles or tours passing through every vertex once and only once.
Currently the problem is considered to be NP-complete, but, nevertheless, two polynomial time deterministic algorithms solving it are known so far. The first one which uses a linear program model with $O(n^9)$ variables and $O(n^7)$ constraints was proposed by Diaby (2007). The second one that uses GFA was suggested by Voinov and Pya Arnqvist (2021).
The algorithm of "tsp_solver" (see Voinov and Pya Arnqvist (2021).p.21) solves both symmetric and asymmetric traveling salesman problems (sTSPs and aTSPs). It is based on solutions of the 0-1 linear Diophantine equation with $n(n-1)$ positive integer elements of a cost/distance matrix. The algorithm finds solutions that represent Hamiltonian cycles of minimal length. For a sTSP the same cycles can be obtained by solving 0-1 equations with $n(n-1)/2$ coefficients of, say, upper or lower triangular elements of the cost/distance matrix. Since the dimension of a corresponding equation is two times less than that for an aTSP, the computing time for a sTSP will be much less than that for the aTSP. Applying this approach the solution of sTSP can be done by the following algorithm:
Alg.2.
Step 1. Solve a corresponding assignment problem for the matrix C to obtain a lower bound (Lb) on the value of the optimal solution. Apply the "cheapest_insertion" algorithm to get an upper bound (Lb).
Step 2. Construct all solutions on exactly n parts for the equation (2.9) with $B=Lb$ and coefficients that are entries of the lower triangular part of C.
Step 3. Select solutions that satisfy conditions of the symmetric HC problem.
Step 4. If there is a solution for the Lb, then stop. Otherwise increase the Lb by one and go to step 2. Repeat until the Ub is reached.
Exp.5.
For all $n \in [3,4,\dots,10]$ two variants of the experiment were performed: one with $B=100$ and the second one with $B \in [24,26,\dots,38]$. 17,800 random instances (12,300 for variant 1, and 5,500 for

the second one) were generated using the R-command “sample(1:*B*,*n*n*,replace=TRUE)”. Times (in sec.) needed to perform the Alg.2 were estimated by the R-command “microbenchmark()”.Numerical results of the simulations are provided in Table 5 (Mt7 for $B = 100$, Mt8 for $B \in [24,26,\ldots,38]$) and Figures 9 and 10 below.

Table 5.

n	*p=n+50n(n+1)*	Mt7		*n*	*B*	*p=n+Bn(n+1)/2*	Mt8
3	603	0.0027605		3	24	147	0.0028344
4	1004	0.0263374		4	26	264	0.0103420
5	1505	0.1098751		5	28	425	0.0392810
6	2106	1.3809700		6	30	636	0.8620258
7	2807	10.627532		7	32	903	5.3806550
8	3608	114.97300		8	34	1232	53.621300
9	4509	958.17140		9	36	1629	487.22540
10	5510	12247.590		10	38	2100	10432.820

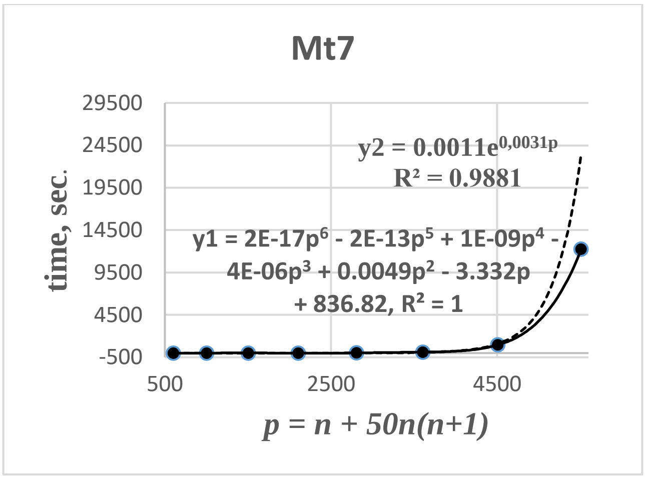


Fig. 9

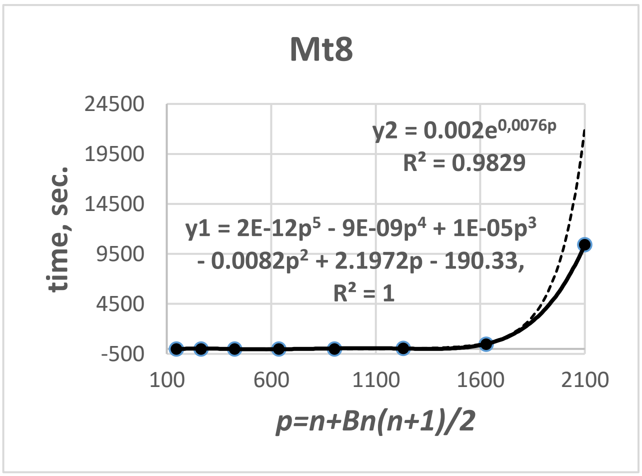


Fig. 10

From these figures we see that the multiple R^2 of polynomial fits (solid lines) is one for both approaches. At the same time for the exponential fits (dashed lines) the values of R^2 are noticeably less than one. Fig.9 corresponds to the restricted **(**$B = 100$) versions of sTSPs. One sees that the computing times are bounded above by the degree 6 polynomial of p. *Observation 4.2* of Garey and Johnson (1979), p.95 states: “if *Π* is NP-complete in the strong sense, then *Π* cannot be solved by a pseudo-polynomial time algorithm unless P=NP”. Thus the experimental results Mt7 are in favor of P=NP, and hence sTSPs are in P. Results Mt8 for the unrestricted version (see Fig.10) confirm this conclusion, since the computing times are bounded above by the degree 5 polynomial of p.

Remark 7.1. Section 6 provides an experimental proof of the HC problem polynomial solvability. Since there is a polynomial time transformation from HC to TSP (ibid, p. 56), one immediately concludes that the sTSP is in P.

Remark 7.2. For the symmetric graph in Fig.8 (Remark 6.1) Alg.2 gives the optimal tour **3 1 6 9 4 2 5 8 7** with the length **226**.

Remark 7.3. Alg.2 produces the same optimal tours for problems stsp81-83 of Diaby (2007), p. 40 thus conferming the correctness of his polynomial-sized linear programming formulation of the TSP.
Remark 7.4. Alg.2 enumerates all existing optimal solutions for the sTSP. Two numerical examples with 6 and 2 optimal solutions are given for illustration in Appendix 2. They can be used as test instances for newly developed algorithms.

8. Discussion and conclusions

Cook (2000), p.7 noted that "the obvious way (to prove that P=NP) is to exhibit a polynomial-time algorithm for 3-SAT or **one** of the other thousand or so known NP-complete problems". Since $P \subseteq NP$ (ibid, p.2), the same way can be used for problems in P. This means that whenever you have a polynomial-time algorithm for **one** of the problems in P (say, L_2), then, if there is a polynomial-time reduction from L_1 to L_2 (*Lemma* 2.1 of Garey and Johnson (1979), p.34), this algorithm can be transformed into a corresponding polynomial-time algorithm for any problem of interest (say, $L_1 \in P$). This reduction will be denoted as $L_1 \Longrightarrow L_2$. Garey and Johnson (1979), p.34 noted that for the class P this relation is *polynomially equivalent* to $L_2 \Longrightarrow L_1$, i.e. $L_1 \Longrightarrow L_2 \Leftrightarrow L_2 \Longrightarrow L_1$. Assume, e.g., that you have a polynomial-time algorithm for Partition and want to transform it to a polynomial time algorithm for a TSP. Figure 11, which is a small part of the general class NP picture, shows that to do this you have to pass 5 transitive reductions. This problem can be simplified if you will use instead a polynomial-time algorithm for the HC problem.

$$\begin{tikzcd}
 & & \textbf{ILP} & \\
\textbf{Partition} \arrow[d, Rightarrow, "\text{(p.223)}"] & \text{3DM} \arrow[l, Rightarrow, "\text{(p.223)}"'] \arrow[d, Rightarrow, "\text{(p.224)}"] & \text{3SAT} \arrow[l, Rightarrow, "\text{(p.221)}"'] \arrow[u, Rightarrow, "\text{(p.245)}"'] \arrow[d, Rightarrow, "\text{(p.190)}"] & \text{SAT} \arrow[l, Rightarrow, "\text{(p.259)}"'] \\
\text{Subset sum} \arrow[d, Rightarrow, "\text{(p.247)}"] & \textbf{N3DM} \arrow[d, Rightarrow, "\text{(p.224)}"] & \textbf{VC} \arrow[r, Rightarrow, "\text{(p.194)}"] \arrow[d, Rightarrow, "\text{(p.199)}"] & \textbf{Clique} \\
\text{Integer knapsack} & \textbf{3-Partition} \arrow[d, Rightarrow, "\text{(p.226)}"] & \textbf{HC} \arrow[d, Rightarrow, "\text{(p.211)}"] & \\
 & \text{Bin packing} & \textbf{TSP} &
\end{tikzcd}$$

Fig. 11. A part of the class NP picture. Problems proved to be in P are boldfaced. Arrows show directions of polynomial-time reductions, proofs of which are cited on presented in parentheses pages of (ibid, Chapter 7).

Fig.11 shows that at least one polynomial-time algorithm of 14 NP problems can be converted to a corresponding polynomial-time algorithm for the rest 13 problems. This is possible due to the polynomial equivalence$L_1 \Longrightarrow L_2 \Leftrightarrow L_2 \Longrightarrow L_1$. Correctness of this identity is confirmed, e.g., by two experimentally proved relations: 3-Partition ⇔ N3DM and HC ⇔ TSP. Thus all 14 NP problems in Fig.11 are in P. The same is true for the whole huge NP class of problems. Summarizing all the above one may conclude that both the class NP and the class P exist and P=NP.

References

Cook, S. (2005). *The P versus NP problem.* https://www.claymath.org, accessed on Jan. 10, 2023.

Diaby, M. (2007). The Traveling Salesman Problem: A Linear Programming Formulation. *WSEAS Transactions on Math, 6(*6), 745-754. arxiv.org/ftp/cs/papers/0609/0609005.pdf.

Garey, M. & Johnson, D. (1978). “Strong” NP-Completeness Results: Motivation, Examples, and Implications. *Journal of ACM*, *25*(3), 499-508.

Garey, M. & Johnson, D. (1979). Computers and Intractability: A Guide to the theory of NP-Completeness, W.H.Freeman & Co., NY.

Kyritis, K. (2021). Review of the solutions of the Clay Millennium problem about P≠NP=EXPTIME. *World J. of Research and Review, 13*(3), 21-26. https://doi.org/10.31871/WJRR.13.3.8.

Laporte, G. (1992). The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European J. of Operational Research, 59*, 231-247. https://doi.org/10.1016/0377-2217(92)90138-Y.

Müeller, M. (2020). Polynomial Exact-3-SAT-Solving Algorithm, *International Journal of Engineering Technologies*, *9*(3), 1-55. https://doi:10.14419/ijet.v9i3.30749.

Nijenhuis, A. & Wilf, H. (1978). *Combinatorial Algorithms for Computers and Calculators*. NY; Academic Press.

Voinov, V. & Nikulin, M. (1995). Generating functions, problems of additive number theory, and some statistical applications. *Romanian J. of Pure and Applied Mathematics*, *40,* 107-147.

Voinov, V. & Nikulin, M. (1997). On a subset sum algorithm and its probabilistic and other applications. Balakrishnan, N, ed.- *Advances in Combinatorial Methods and Applications to Probability and Statistics*. Boston: Birkhäuser, 153-163.

Voinov, V. (2017). A note on the intractability of partitions, knapsack, subset sum and related problems. *Mathematical Journal* (ISSN 1682-0525), *17*(4), 13-24.

Voinov, V., Pya Arnqvist, N. & Voinov, Y. (2018). Polynomial in time nonnegative integer solutions of knapsack and similar problems in R: P=NP? *Mathematical Journal*, *18*(2), 47-58.

Voinov, V. (2019). A note on serious arguments in favor of equality P=NP. *18th ASMDA Conf. Proceedings, 11-14 June 2019, Florence, Italy*, 799-809.

Voinov, V. & Rahmanov, M. (2020). 2-, 3-, and 4-Partition Problems and their Relation to the Equality P=NP. *Central Asia Business Journal*, *11*(2), 34-46.

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.

Voinov, V. (2022). An algorithm for deriving classical and Generalized Frobenius numbers: applications in tiling and storing. *Central Asia Business Journal, 13*(1), 23-35.

Wen-Qi, D. (2012). *A constructive algorithm to prove P=NP*. http://arxiv.org/abs/1208.0542, accessed on Jan. 10, 2023.

Woeginger, G. (2016). *The P versus NP page*. https://www.win.tue.nl/~gwoegi/P-versus-NP.htm, accessed on Jan. 10, 2023.

Zeilberger, D. (2009.) *Research Announcement: A Computer-Generated Proof that P=NP.* http://www.math.rutgers.edu/~zeilberg/mamarimPDF/pnp.pdf, accessed on Jan. 10, 2023.

Author Bio: In 1964, Vassilly Voinov graduated from the Tomsk State University in the former USSR. In 1989 he has defended a doctoral thesis in mathematical statistics in physics at the Joint Institute for Nuclear Research (Dubna, Moscow region). He is a leading author of five books in mathematical statistics and one in experimental nuclear physics. Three of them have been published by Kluwer and one by Academic Publishers. Since 1998 he has been teaching 12 courses related to mathematical statistics at KIMEP University for undergraduate and graduate students. He was a member of the American Statistical Association and the American Mathematical Society. He has published about 120 research papers. He coauthored also one invention, and three published contributions in mathematical software. Retired in 2021.

Appendix 1. R-script “two_parts”

```
library(combinat); library(nilde); library(microbenchmark)
genSample<-function(M,n,D){
x<-array(0,dim=n); x<-sample(1:10,n,replace=FALSE)
y<-sum(x); z1<-y==2*D
while(z1!=TRUE){
x<-sample(1:10,n,replace=FALSE); y<-sum(x); z1<-y==2*D
}
xx<-sort(x); d2<-n; d4<-M; d6<-D; d1<-list(xx,d2,d4,d6,y); return(d1)
}
k<-50
Tnat<-rep(NA,k); Tcomb<-rep(NA,k); Tb<-proc.time()[3]
set.seed(853) # set desired seed
for(i in 1:k){
ns<-genSample(2,5,13) # specify desired instances
d4<-ns[[3]] # M
d6<-ns[[4]];d7<-ns[[1]];d2<-ns[[2]];d8<-ns[[5]]
g<-get.subsetsum(a=d7,n=d6,M=d2,problem="subsetsum01")
if(g$p.n>=2){
print(d7) # A
print(sum(d7))
T30<-microbenchmark(g<-get.subsetsum(a=d7,n=d6,M=d2,problem="subsetsum01"),times=1L)
T301<-T30$time/1000000000; b10<-g$solutions
print(b10) # solutions by (1.10)
print(d6) # D
b11<-microbenchmark(Solnat<-as.matrix(b10[,colSums(b10)>=2]),times=1L)
T302<-b11$time/1000000000; Tnat[i]<-T301+T302
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
print(Solcomb) # first part (A')of the solution by ICA
a2<-microbenchmark(a4<-combn(x=d7,m=(d4+1),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) # second part (A-A') of the solution by ICA
}; }
Te<-proc.time()[3]; print(Te-Tb)
Tnat<-Tnat[!is.na(Tnat)]
k10<-length(Tnat);k10
mtnat<-mean(Tnat); 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. Examples of several optimal solutions

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]  999   12   25   27   10   18   24   13   22    21
 [2,]   12  999   24   25    9   37   14   30   16    24
 [3,]   25   24  999   20   11   32   19   25   23    18
 [4,]   27   25   20  999   14   16   32   24   18    13
 [5,]   10    9   11   14  999   24   26   23   16    28
 [6,]   18   37   32   16   24  999   22   17   25    34
 [7,]   24   14   19   32   26   22  999   15   27    23
 [8,]   13   30   25   24   23   17   15  999   17    16
 [9,]   22   16   23   18   16   25   27   17  999    32
[10,]   21   24   18   13   28   34   23   16   32   999
$tour.1
 [1]  6  1  8  7  2  9  5  3 10  4
$tour.2
 [1]  2  1  6  4 10  3  5  9  8  7
$tour.3
 [1]  5  1  6  8  7  2  9  4 10  3
$tour.4
 [1]  5  1  6  4 10  8  9  2  7  3
$tour.5
 [1]  5  1  8  9  2  7  6  4 10  3
$tour.6
 [1]  2  1  5  3 10  4  6  7  8  9
[1] 150 # Tours' length
[1] 153 # Upper bound
[1] 146 # Lower bound
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]  999   72   91  126  111   99  109   45   94
[2,]   72  999  116   83   77   72   78  170  101
[3,]   91  116  999  109   20   65   29   71  141
[4,]  126   83  109  999   76  123  159   90  155
[5,]  111   77   20   76  999   35   40   96   60
[6,]   99   72   65  123   35  999   81   27   92
[7,]  109   78   29  159   40   81  999  109  127
[8,]   45  170   71   90   96   27  109  999  146
[9,]   94  101  141  155   60   92  127  146  999
$tour.1
[1] 2 1 9 5 3 7 6 8 4
$tour.2
[1] 8 1 9 2 4 5 3 7 6
[1] 556 # Tours' length
[1] 560 # Upper bound
[1] 485 # Lower bound
```