World Applied Sciences Journal 24 (8): 1072-1083, 2013
ISSN 1818-4952
© IDOSI Publications, 2013
DOI: 10.5829/idosi.wasj.2013.24.08.13256

The Lower Border of Complexity of Algorithm of the Elementary NP-Complete Task (The Most Condensed Version)

Rustem Chingizovich Valeyev
NPO CKTI, Joint-Stock Company I.I.Polzunov Scientific and Fevelopmen
Association on Research and Design of Power Equipment, St-Petersburg, Russia

Submitted: Aug 5, 2013; **Accepted:** Sep 4, 2013; **Published:** Sep 4, 2013

Abstract: Complexity of any mass task (and, consequently, the minimal computational complexity of algorithm) is proportional to minimal quantity of operations made by algorithm which are necessary and sufficient for reception of the required answer of any individual case of this task. Complexity of the most elementary NP-complete task TSP ("Traveling Salesman Problem") it is estimated proceeding from features of a structure of its goal function (power of this set, quantity of domains of this dependence etc.). It is shown, that the most effective algorithm of the decision of the mass task TSP is exponential algorithm "exhaustive search". It concerns also to all other tasks of class NP. In other words, the class P doesn't coincide with class NP.

Key words: Complexity · NP-complete task · Traveling Salesman Problem · Problem P vs NP · Primes

INTRODUCTION

For last 40 years hundreds of mass NP-complete tasks which have the vital applied importance are found out. The whole meaning of arisen questions consists in the following. Let for some discrete task it is not possible to design polynomial algorithm despite of serious efforts of many mathematicians. In what the reason of these failures? In that, that such algorithm does not exist in nature, or polynomial algorithm exists but to find him very difficultly? This unsolved till now fundamental problem got a name "problem P versus NP" [1, 2, 3].

Does polynomial algorithm of solution at least one NP-complete task is exist? If yes, then it would be possible successfully and quickly to solve any tasks from class NP by means of their reduction to this successful NP-complete task. There is a large variety of the points of view on this question (for example, [4], [5], [6]).

The full version of this article: http://www.rst-valeyev.ru/page.php?id=spec.

Preliminary Remarks: How dimension hereinafter understands a certain parameter of a mass task which anyhow defines or quantity of examined objects, or - quantity of actions which essentially should be present at algorithm of the solution of a task. Algorithms are understood only as algorithms, which allow to receive exact answers of tasks. In square brackets there are words which are clear from a context and can be missed.

It's often that there is no difference between ñoncepts "variable" and "value of variable" in mathematic practice. In general case such unification of comprehension isn't fair. That's why here and than such comprehensions as "variable" and "value of variable" will be strongly differ from each other and mean respectively "set which members define or change any dependence" and "separate member of such set".

Even more rarely taken note of the fact that such "separate set members, which is a variable" can be defined by more than one parameter. Hereinafter let's assume that every i-th set member of any set is completely determined by the parameter P_0 (which has a only one value p_0 as "this set member belongs to a given set") and by the finite subset of parameters.

$$X = \{x_1, x_2, x_3, ...\} \text{ or } X = \{x_1, x_2, x_3, ..., x_n\}$$
$$x_i = \{P_0, P_{i1}, P_{i2}, ..., P_{ij}, ..., P_{ik}\} = \{p_0, p_{i1}, p_{i2}, ..., p_{ij}, ..., p_{ik}\}$$

Uncertainty [about answer] - it's a fact that there are two or more options for answer in the task.

Corresponding Author: Rustem Chingizovich Valeyev, NPO CKTI Joint-Stock Company I.I.Polzunov Scientific and Fevelopment Association on Research and Design of Power Equipment, Atamanskaya Str., 3/6, 191167, St-Petersburg, Russia.

Definitions: N-dimensional space of a task - the Descartes product of N sets. Each of these N sets is the set of all the values of one of the N (where $N \geq n$) variables that exist in the task (n is the quantity of unknowns).

"Oracle", "from the point of view of the oracle" is a paraphrases of a short formal wording of "objective reality existing in the real world. Similarly, "customer", "developer" and "user" of algorithm (all these roles can be united the term "detached onlooker") – is convenient paraphrase of the wording "a man trying to solve a task".

Possible answer [of a task] - any point of N-dimensional space of this task.

Forbidden answer - any point of N-dimensional space which does not belong to any of limitations declared by the developer of algorithm.

Allowable answer - any point of N-dimensional space which is not the forbidden answer.

Exact answer - a subset of allowable answers; each set member of this subset completely satisfies to all those conditions which contain in initial data of this task.

Set of FPR_i of objective virtual limitations of the mass task T_i – the subset of all necessary and sufficient parameters, which in terms of the oracle should be stipulated in algorithm in order to completely define the exact answer of T_i.

Obligatory value of parameter of a task T_i – value of any member of set FPR_i.

Correct algorithm – an algorithm applied to solve the tasks for which it is destined, from the point of view of the oracle.

Exhaustive search – algorithm using detection of any and all set members of the set and determining the value (s) of each member of this set.

Set of subjective limitations of FPA_i of a task T_i – the set members of FPR_i which values can specify the developer of algorithm.

Set of obligatory operations OPR_i of a mass task T_i – set of all necessary and sufficient (from point of view of oracle) actions without fulfilment of which the correct algorithm of the decision of T_i is incompetent.

Set of prospective operations OPA_i of mass task T_i – a subset of all obligatory operations which the developer of algorithm can provide.

Criterion of competence of algorithm ACr_i – the expression defining possibility of receiving by means of this algorithm exact answers of the mass task T_i.

If $FPA_i = FPR_i$ (i.e. these two sets consist of the same elements) and simultaneously $OPA_i = OPR_i$, then $ACr_i = 1$.

$$Acr_i = 0.5 \times (|FPA_i|/|FPR_i| + |OPA_i|/|OPR_i|)$$

Competent algorithm [of task T_i] - correct algorithm which $ACr_i = 1$.

Axiom 1: Any mass task T_i is completely and uniquely defined by the set of objective virtual limitations of FPR_i without the indication of values?? which define individual cases of T_i.

More than obvious disadvantages of using definitions of "polynomial" and "exponential" as a criterion for the feasibility of something in real world sometimes brings mess in the term "efficient algorithm". Therefore, the concept of "accessibility" are using for further convenience, simplicity and unambiguity of the question of the efficiency of algorithms (see also [8], [9]).

Definition: Accessibility of parameter A - opportunity for developer to realize application for all possible values of parameter A in order to get an exact answer for any of the individual cases of the mass task.

Effective can be only such algorithm, which the process of obtaining the required answer fit into the frameworks of astronomical time within which the receipt of this answer still has at least some practical sense.

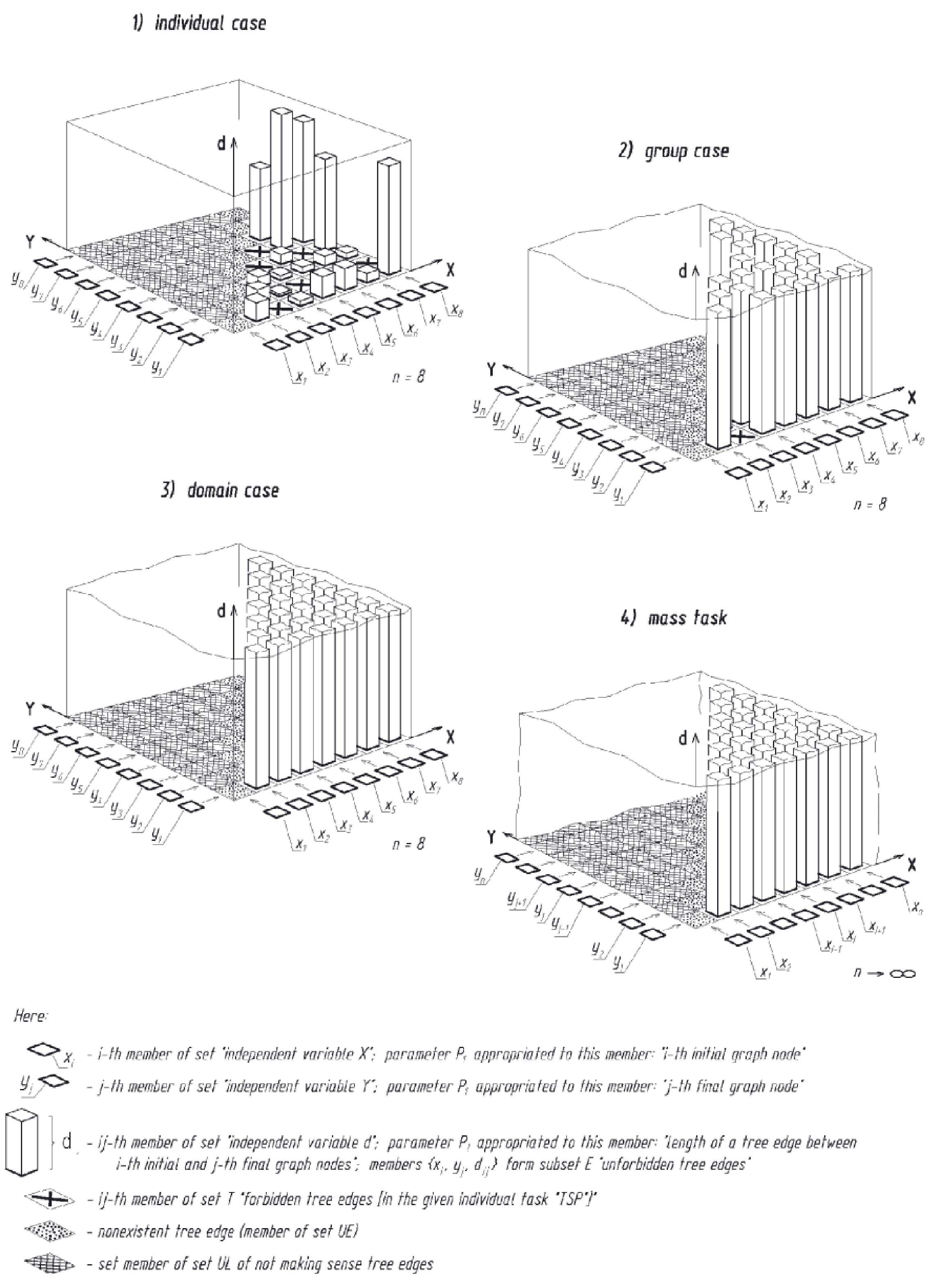
In addition to well-known terms of "mass task" and "individual [case of] task" relation to some of the tasks on discrete structures it is convenient to use concepts "group [case of] task" and "domain [case of] task", "domain" algorithm", "mass algorithm" etc (Fig. 1).

Consequently, the complexity of the algorithm ALG_i for solving of some mass task T_i is determined by three parameters:

- Quantity of obligatory values of parameters which such algorithm must process to get the exact answer of any individual case of T_i;
- Quantity of obligatory operations which such algorithm must commit to get an exact answer of any individual case of T_i;
- Power of set "mass algorithm"

$$|ALG_i| = |\{ ALG_{i1}, ALG_{i2}, ALG_{i3}, ...\}|$$

For the analysis of structure of the task TSP it is convenient to consider the measuring parameters (quantitative sets) separately from the topological properties (matrixes).


1) individual case
2) group case
3) domain case
4) mass task
d
X
Y
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
y_n
x_n
x_{i-1} x_i x_{i+1}
y_{i-1} y_i y_{i+1}
n = 8
n → ∞

Here:

x_i - i-th member of set "independent variable X"; parameter P_r appropriated to this member: "i-th initial graph node"

y_j - j-th member of set "independent variable Y"; parameter P_r appropriated to this member: "j-th final graph node"

d - ij-th member of set "independent variable d"; parameter P_r appropriated to this member: "length of a tree edge between i-th initial and j-th final graph nodes"; members $\{x_i, y_j, d_{ij}\}$ form subset E "unforbidden tree edges"

- ij-th member of set T "forbidden tree edges (in the given individual task "TSP")"

- nonexistent tree edge (member of set UE)

- set member of set UL of not making sense tree edges

Fig. 1: Individual, group and domain cases of the mass task TSP for n = 8 and the mass task TSP

About Independent Variables: How often, a set A "independent variable" is some series of numbers or the set of quantitative values of some function. But at many tasks on discrete structures this set can be the set, where parameter P_{1i} of an element a_i is not quantitative value (an example – the task TSP).

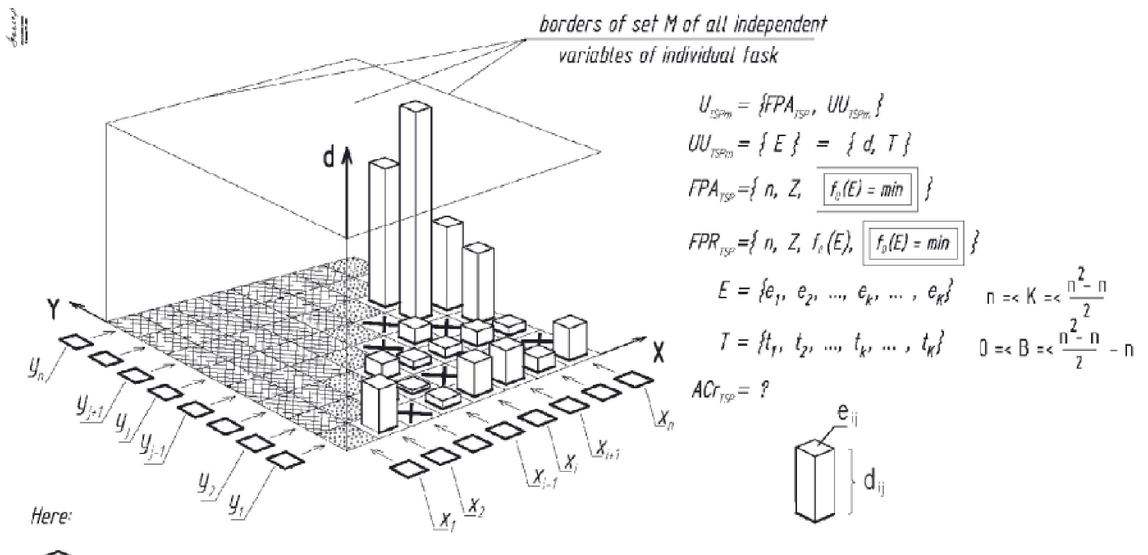
And, in theory, nothing prevents the fact that the value of parameter P_{1i} of element a_i in a set A subject to more than one functional dependency.

Consequently, the situation when as the parameter of values of variables numbers from any numerical series appear, values of any functions, etc., represents only special case of a special case.

Independent Variables and Arguments of Goal Function in the Task TSP: In the task TSP independent variables declared by the user and those parameters, which are arguments of goal function is not same (otherwise the problem P vs NP would not exist). Moreover, they are connected (or, perhaps, are divided) by combinatory dependence. This dependence a priori does not submit to the compact quantitative (analytical, functional, calculated) laws and leaves very few chances of creation of polynomial algorithms.

Hereinafter under arguments of goal function of considered (individual, group, domain, mass) task TSP are understood all those arguments of this dependence, which the developer is obliged to set for the successful decision of all without exception of the individual cases forming considered task.

Theorem 1: All quantitative values except for quantity of graph nodes, declared by the user of algorithm in initial data of any individual case of the task TSP (i.e. values of independent variables), represent functions which are differed from each other and are not depended on each other.



$U_{TSPm} = \{FPA_{TSP},\ UU_{TSPm}\}$

$UU_{TSPm} = \{E\} = \{d, T\}$

$FPA_{TSP} = \{n,\ Z,\ \boxed{\boxed{f_0(E) = min}}\ \}$

$FPR_{TSP} = \{n,\ Z,\ f_t(E),\ \boxed{\boxed{f_0(E) = min}}\ \}$

$E = \{e_1,\ e_2,\ \ldots,\ e_k,\ \ldots,\ e_K\}$ $\quad n =< K =< \frac{n^2 - n}{2}$

$T = \{t_1,\ t_2,\ \ldots,\ t_k,\ \ldots,\ t_K\}$ $\quad 0 =< B =< \frac{n^2 - n}{2} - n$

$ACr_{TSP} = ?$

Here:

◇ x_i -ij-th element of set "independent variable X"; parameter P_1 appropriated to this element: "i-th initial graph node"

y_j ◇ - j-th element of set "independent variable Y"; parameter P_1 appropriated to this element: "j-th final graph node"

d_{ij} - ij-th element of set "independent variable d"; parameter P_1 appropriated to this element: "length of a tree edge between i-th initial and j-th final graph nodes"

e_{ij} - ij-th set member of the set E "unforbidden tree edges in the given individual task "TSP""; this declared by the user set members are convenient for interpreting as vectors with the beginning in the beginning of coordinates

◇ - ij-th set member of set WZ "matrix of making sense tree edges [in the given individual task "TSP"]"

⊕ - ij-th set member of subset WT "matrix of forbidden tree edges"; ◈ - --"-- of subset WE "matrix of unforbidden tree edges"

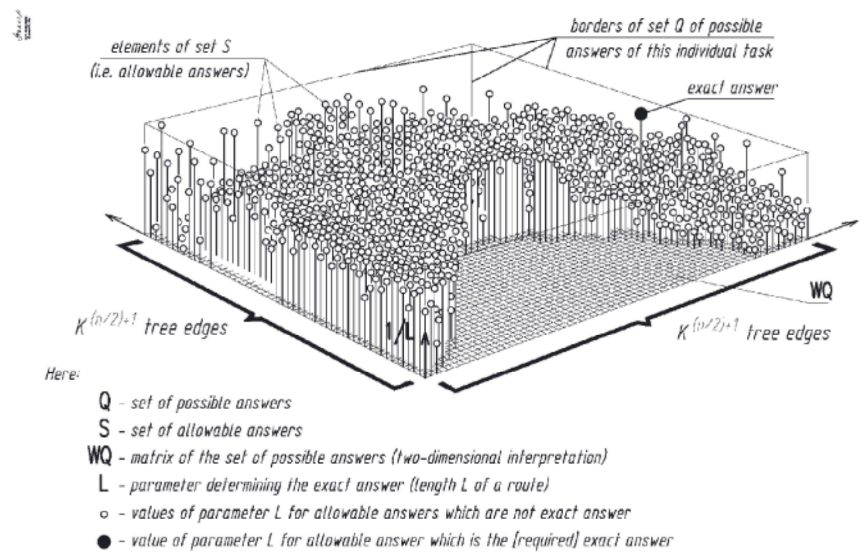
▒ - set member of set WUE "matrix of nonexistent tree edges"

▦ - set member of set WUL "matrix of not making sense tree edges" (edges which have no useful physical sense for algorithm)

f_0 (E) - goal function (the sum of lengths of all tree edges of a route)

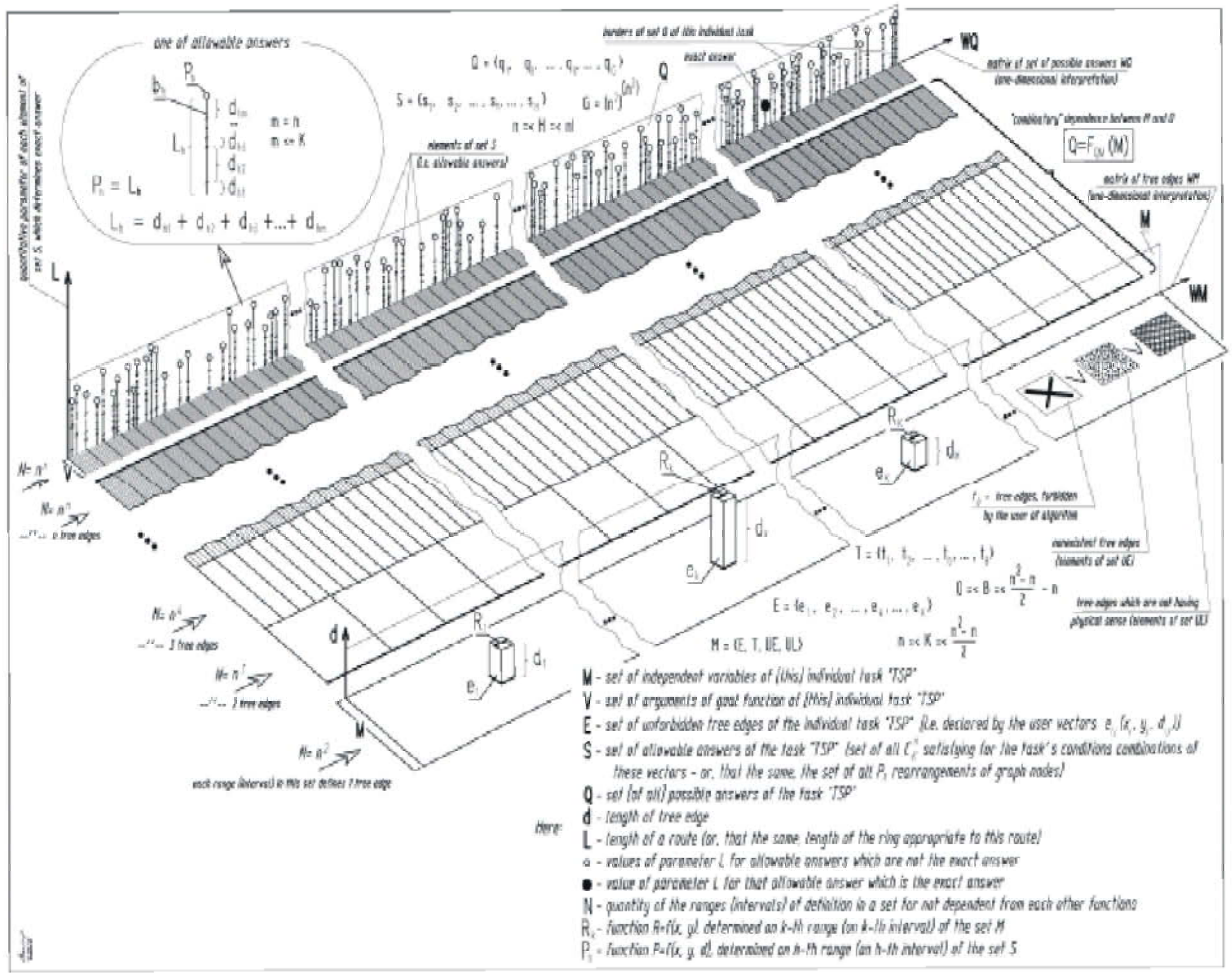
$\boxed{\ }$ - sign of the exact answer (i.e. a condition which reveals the exact answer among other allowable answers)

Fig. 2: Set of all independent varibles in the taks TSP (interpretation as the three-dimensianal ordered space)



elements of set S (i.e. allowable answers)
borders of set Q of possible answers of this individual task
exact answer
WQ
1/L
$K^{(n/2)+1}$ tree edges
$K^{(n/2)+1}$ tree edges
Here:
Q - set of possible answers
S - set of allowable answers
WQ - matrix of the set of possible answers (two-dimensional interpretation)
L - parameter determining the exact answer (length L of a route)
○ - values of parameter L for allowable answers which are not exact answer
● - value of parameter L for allowable answer which is the [required] exact answer

Fig. 3: Set of allowable anwers in the task TSP (interpretation as the three-dimensional ordered space)



one of allowable answers
P_h
b_h
d_{hm}
d_{h3}
d_{h2}
d_{h1}
L_h
m = n
m =< K
$P_h = L_h$
$L_h = d_{h1} + d_{h2} + d_{h3} + ... + d_{hm}$
quantitative parameter of each element of set S, which determines exact answer
L
$Q = (q_1, q_2, ..., q_j, ..., q_G)$
$S = (s_1, s_2, ..., s_h, ..., s_H)$
$G = (n^2)^{(n^2)}$
n =< H =< n!
Q
borders of set Q of this individual task
exact answer
WQ
matrix of set of possible answers WQ (one-dimensional interpretation)
elements of set S (i.e. allowable answers)
"combinatory" dependence between M and Q
$Q = F_{CN}(M)$
matrix of tree edges WM (one-dimensional interpretation)
M
WM
R_k
e_k
d_k
R_2
e_2
d_2
R_1
e_1
d_1
t_{ij} - tree edges, forbidden by the user of algorithm
nonexistent tree edges (elements of set UE)
$T = (t_1, t_2, ..., t_j, ..., t_B)$
$0 =< B =< \frac{n^2 - n}{2} - n$
tree edges which are not having physical sense (elements of set UL)
$E = (e_1, e_2, ..., e_k, ..., e_K)$
M = (E, T, UE, UL)
$n =< K =< \frac{n^2 - n}{2}$
V
N = n^2
N = n^2 ... n tree edges
N = n^2 ... 3 tree edges
N = n^2 ... 2 tree edges
N = n^2
each range (interval) in this set defines 1 tree edge
d
M
M - set of independent variables of (this) individual task "TSP"
V - set of arguments of goal function of (this) individual task "TSP"
E - set of unforbidden tree edges of the individual task "TSP" (i.e. declared by the user vectors e_k (x_i, y_j, d_{ij}))
S - set of allowable answers of the task "TSP" (set of all C_K^n satisfying for the task's conditions combinations of these vectors - or, that the same, the set of all P_h rearrangements of graph nodes)
Q - set (of all) possible answers of the task "TSP"
d - length of tree edge
Here:
L - length of a route (or, that the same, length of the ring appropriate to this route)
○ - values of parameter L for allowable answers which are not the exact answer
● - value of parameter L for that allowable answer which is the exact answer
N - quantity of the ranges (intervals) of definition in a set for not dependent from each other functions
R_k - function R=f(x, y), determined on k-th range (on k-th interval) of the set M
P_h - function P=f(x, y, d), determined on h-th range (on h-th interval) of the set S

Fig. 4: Set of independent variables and set of allowable answers in a task TSP with n graph nodes (interpretation in form of two-dimensional ordered spaces)

Proof:

- The user of algorithm has the right to declare any positive constant (quantitative item d_{ij}) as length of any unforbidden tree edge (topological item, named "(x_i, y_j)").
- Any constant d_{ij} can be considered as the single value of a certain single-valued function of $qw_{ij}(x_i, y_j)$ defined on a point interval (point range) with coordinates (x_i, y_j).
- Determined by the graph nodes unforbidden tree edges in any individual case by definition is not related to each other by any quantitative and/or other functional dependencies.
- Therefore, functions $d_{ij} = qw_{ij}(x_i, y_j)$ in any individual case of the task TSP in general case represent unique (i.e. existing only on its point ranges in considered individual case) dependences.
- Consequently, from the point of view of the oracle in TSP the declaration by the user unforbidden tree edge and the length of d_{ij} of this edge means declaration of one function $d_{ij} = qw_{ij}(x_i, y_j)$ which in general case:
- Differs from any other dependence in this and in any other individual case of TSP;
- Doesn't depend on any other dependence in this and any other individual case of TSP;
- Exists (is declared) only on one point interval (range) with coordinates x_i and y_j (this interval doesn't depend on any other such interval "tree edge" in a n-node graph in this and in any other individual case).

It is easily possible to prove also the following obvious statements:

Theorem 2: The values of independent variables of each individual case of [mass] task TSP are polynomial set of unique (i.e. existing only for this very individual case) different from each other and independed on each other functions.

Theorem 3: The values of independent variables of each group case of TSP are infinity set of existing only for this very group case different from each other and independed on each other functions.

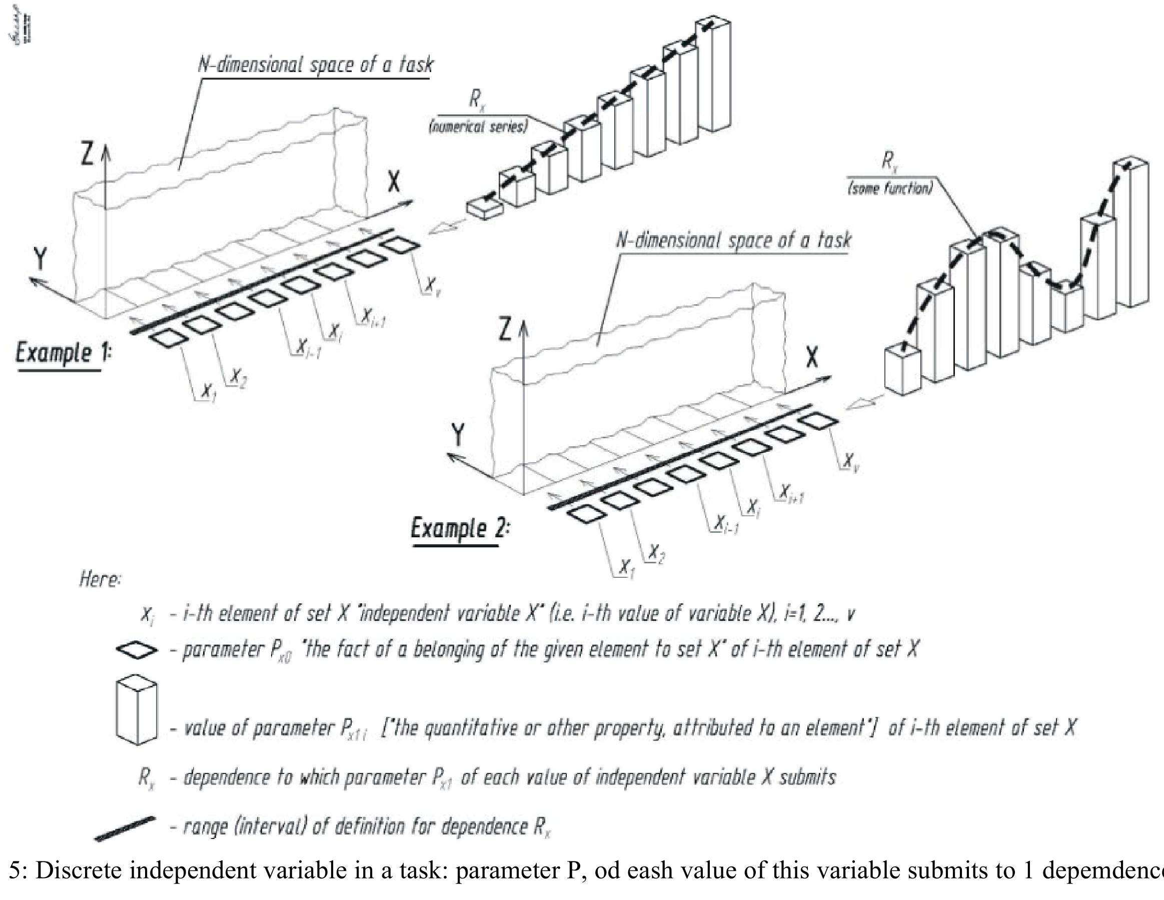

N-dimensional space of a task
Z
X
Y
R_x
(numerical series)
X_1
X_2
X_{i-1}
X_i
X_{i+1}
X_v
Example 1:
N-dimensional space of a task
R_x
(some function)
Z
X
Y
X_1
X_2
X_{i-1}
X_i
X_{i+1}
X_v
Example 2:
Here:
X_i - i-th element of set X "independent variable X" (i.e. i-th value of variable X), i=1, 2..., v
- parameter P_{x0} "the fact of a belonging of the given element to set X" of i-th element of set X
- value of parameter P_{x1i} ["the quantitative or other property, attributed to an element"] of i-th element of set X
R_x - dependence to which parameter P_{x1} of each value of independent variable X submits
- range (interval) of definition for dependence R_x

Fig. 5: Discrete independent variable in a task: parameter P, od eash value of this variable submits to 1 depemdence R_X

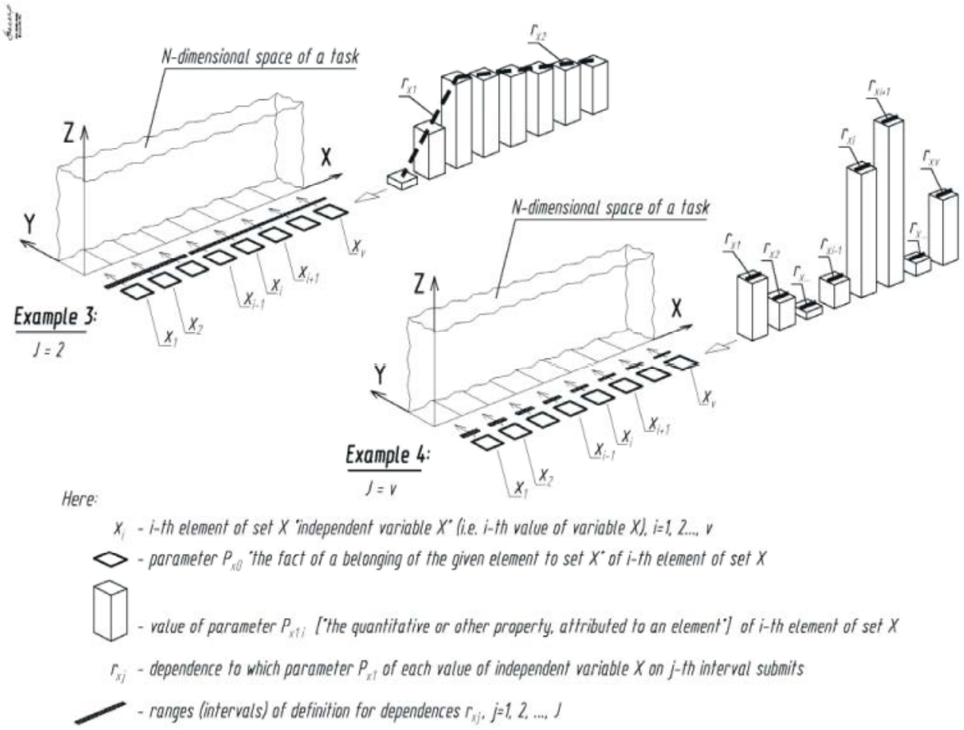

N-dimensional space of a task
Z
X
Y
r_{x1}
r_{x2}
x_1
x_2
x_{i-1}
x_i
x_{i+1}
x_v
Example 3:
J = 2
N-dimensional space of a task
Example 4:
J = v
Here:
x_i - i-th element of set X "independent variable X" (i.e. i-th value of variable X), i=1, 2..., v
- parameter P_{x0} "the fact of a belonging of the given element to set X" of i-th element of set X
- value of parameter P_{x1i} ["the quantitative or other property, attributed to an element"] of i-th element of set X
r_{xj} - dependence to which parameter P_{x1} of each value of independent variable X on j-th interval submits
- ranges (intervals) of definition for dependences r_{xj}, j=1, 2, ..., J

Fig. 6: Discrete independent variable in a task: parameter P, of ech value of this variable not submits to 1 dependence

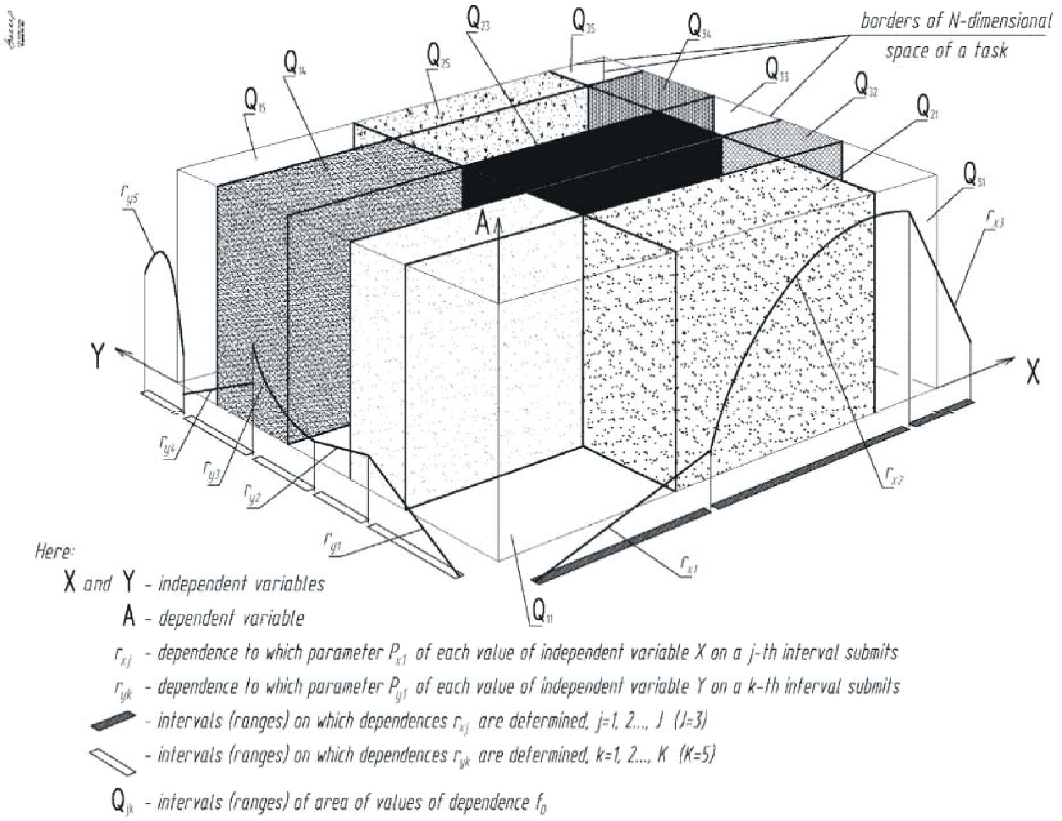

borders of N-dimensional space of a task
Q_{15}
Q_{14}
Q_{25}
Q_{33}
Q_{35}
Q_{34}
Q_{32}
Q_{21}
Q_{31}
Q_{11}
A
Y
X
r_{y5}
r_{y4}
r_{y3}
r_{y2}
r_{y1}
r_{x3}
r_{x2}
r_{x1}
Here:
X and Y - independent variables
A - dependent variable
r_{xj} - dependence to which parameter P_{x1} of each value of independent variable X on a j-th interval submits
r_{yk} - dependence to which parameter P_{y1} of each value of independent variable Y on a k-th interval submits
- intervals (ranges) on which dependences r_{xj} are determined, j=1, 2..., J (J=3)
- intervals (ranges) on which dependences r_{yk} are determined, k=1, 2..., K (K=5)
Q_{jk} - intervals (ranges) of area of values of dependence f_0

Fig. 7: Intervals (ranges) of area of values of dependence f_0 in N-dimensianal space of a task (N=3, independent variables X and Y are determined on 3 and 4 intervals accordingly

Theorem 4: The values of independent variables of each domain case of TSP are infinite set of existing only for this very domain case different from each other and independed on each other functions.

Theorem 5: The values of independent variables of TSP represent the infinite set of existing only for the TSP different from each other and independend on each other functions.

Features of Goal Function of the Task TSP

Theorem 6: In each individual case of the task TSP goal function is the unique (i.e. not repeating in one other individual case of TSP) finite set consisting of exponential quantity of different from each other and independed on each other functions:

$L_a = fw_a(E_a) = fw_a(e_{a1}, e_{a2}, ..., e_{ak}, ..., e_{aK}) = \{fw_{a1}(E_a), fw_{a2}(E_a), ..., fw_{ah}(E_a), ..., fw_{aH}(E_a)\}$

$n \le |\{fw_{a1}(E_a), fw_{a2}(E_a), ..., fw_{ah}(E_a), ..., fw_{aH}(E_a)\}| \le n!, n \to +\infty$

Here: $fw_a(E_a)$ – goal function (minimum of this dependence is the required exact answer).

E_a – independent variables (finite set declared by the user of unforbidden tree edges of a-th individual case)
K – power of the set E_a, $n \le K \le (n^2-n)/2$
n – quantity of graph nodes
H – quantity of allowable answers, $n \le H \le n!$

$d_{ij}(x_i, y_j)$ – the function defined on a point interval with limits of range [with interval's borders] (x_i, y_j)

x_i – i-th initial graph node
y_j – j-th final graph node
$fw_{ah}(\,)$ – h-th argument of goal function
d_{ijhm} – m-th argument of function $fw_{ah}(\,)$

Proof:

- The set E_a can be viewed as a matrix of independent variables of WE_a, each k-th point of that matrix is not forbidden and correspondence to length of this tree edge.
- The set S_a can be viewed as a matrix of WS_a of allowable answers. Each h-th matrix's element is delivered in uniquely compliance length of L_{ah} of passing only one time through each of n graph's nodes of h-th closed route (is unique combination of n not repeating in h-th closed route unforbidden tree edges).
- WS_a represents a set of all swaps of P_n of n-node graph

$|S_a| = |WS_a| = H = P_n = n!$

- As it is known from combinatorial theory, dependence of $S_a=F_{SE}(E_a)$ isn't the functional measuring dependence.
- According to the theorem 2 set members of E_a is unique functions, which are differed from each other and are not depended on each other

$qw_a = \{d_{ij}(x_i, y_j) / i = 1, 2, 3, ..., n; j = 1, 2, 3, ..., n; (x_i, y_j) \in WE_a\}$
$n < |qw_a| \le (n^2-n)/2$

- It doesn't require the proof that any combinations of unique elements of the initial set always form unique combinations. Therefore each of H set members of S_a by definition is unique, not repeating in S_a, set consisting of n not repeating set members of E_a.
- The exponential quantity of items, each of which represents polynomial value, is exponential value.

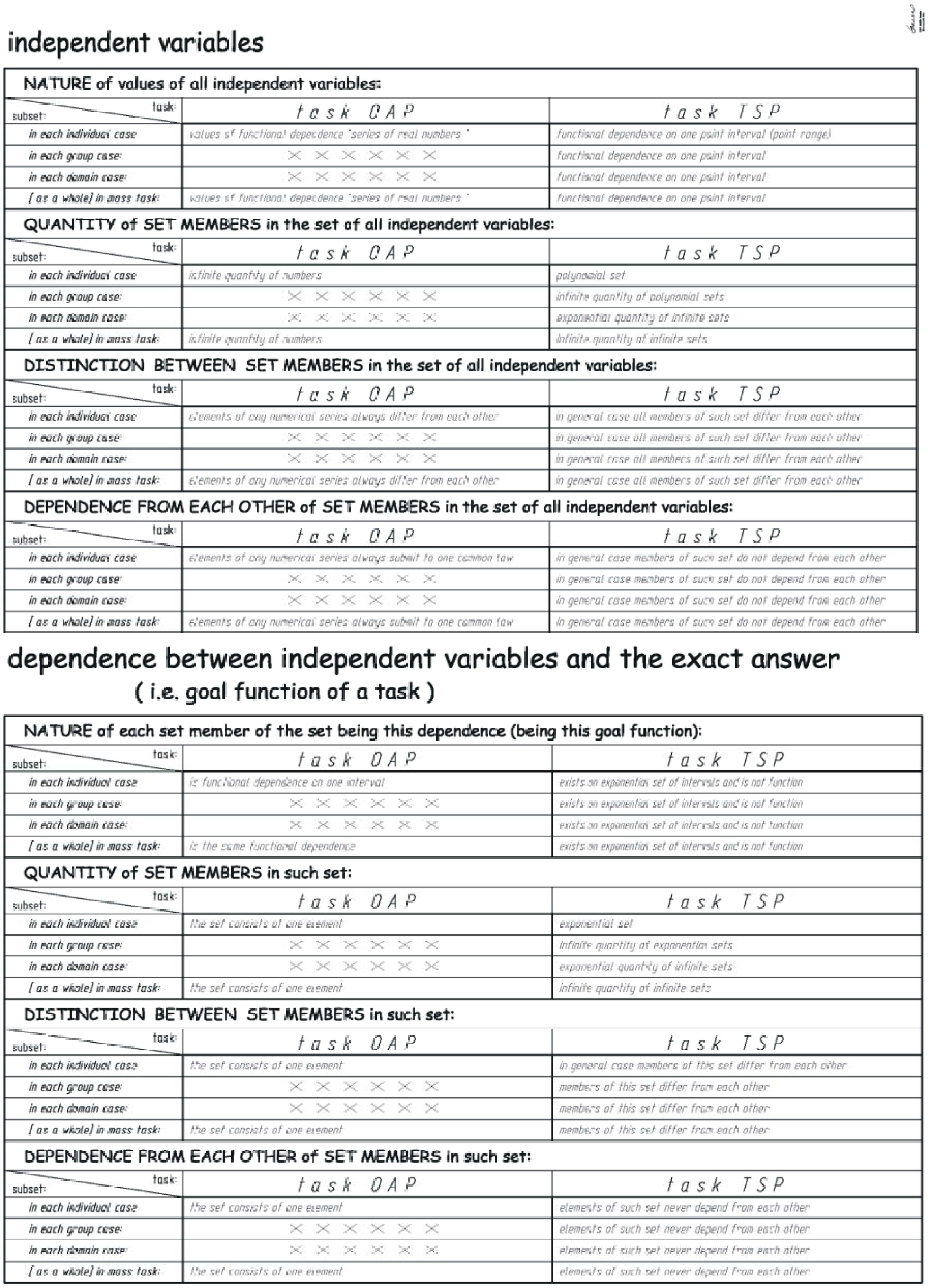
$L_{ah} = fw_{ah}(\,), h = 1, 2, 3, ..., n!$

- Therefore, goal function of $L_a=fw_a(E_a)$ is unique exponential set of dependences differing from each other and are not depending on each other.

Each member of set $fw_a(E_a)$ "goal function" has the unique set of values of its arguments. Thus, any individual case of TSP has polynomial set of values of independent variables (they declare by user) and exponential set of values of arguments of goal function (they are created by laws of combination theory and should be taken beforehand into account in algorithm by the developer).

In the similar way it is possible to prove statements for group, domain and mass task TSP:

Theorem 7: In each group case of TSP dependence between a set of independent variables and exact answers of each individual case of this group case (i.e. goal function of this group case) is not repeating in other group cases infinite set of different from each other and independed on each other functions,



independent variables

NATURE of values of all independent variables:

task: / subset:	task OAP	task TSP
in each individual case	values of functional dependence "series of real numbers"	functional dependence on one point interval (point range)
in each group case:	× × × × × ×	functional dependence on one point interval
in each domain case:	× × × × × ×	functional dependence on one point interval
[as a whole] in mass task:	values of functional dependence "series of real numbers"	functional dependence on one point interval

QUANTITY of SET MEMBERS in the set of all independent variables:

task: / subset:	task OAP	task TSP
in each individual case	infinite quantity of numbers	polynomial set
in each group case:	× × × × × ×	infinite quantity of polynomial sets
in each domain case:	× × × × × ×	exponential quantity of infinite sets
[as a whole] in mass task:	infinite quantity of numbers	infinite quantity of infinite sets

DISTINCTION BETWEEN SET MEMBERS in the set of all independent variables:

task: / subset:	task OAP	task TSP
in each individual case	elements of any numerical series always differ from each other	in general case all members of such set differ from each other
in each group case:	× × × × × ×	in general case all members of such set differ from each other
in each domain case:	× × × × × ×	in general case all members of such set differ from each other
[as a whole] in mass task:	elements of any numerical series always differ from each other	in general case all members of such set differ from each other

DEPENDENCE FROM EACH OTHER of SET MEMBERS in the set of all independent variables:

task: / subset:	task OAP	task TSP
in each individual case	elements of any numerical series always submit to one common law	in general case members of such set do not depend from each other
in each group case:	× × × × × ×	in general case members of such set do not depend from each other
in each domain case:	× × × × × ×	in general case members of such set do not depend from each other
[as a whole] in mass task:	elements of any numerical series always submit to one common law	in general case members of such set do not depend from each other

dependence between independent variables and the exact answer
(i.e. goal function of a task)

NATURE of each set member of the set being this dependence (being this goal function):

task: / subset:	task OAP	task TSP
in each individual case	is functional dependence on one interval	exists on exponential set of intervals and is not function
in each group case:	× × × × × ×	exists on exponential set of intervals and is not function
in each domain case:	× × × × × ×	exists on exponential set of intervals and is not function
[as a whole] in mass task:	is the same functional dependence	exists on exponential set of intervals and is not function

QUANTITY of SET MEMBERS in such set:

task: / subset:	task OAP	task TSP
in each individual case	the set consists of one element	exponential set
in each group case:	× × × × × ×	infinite quantity of exponential sets
in each domain case:	× × × × × ×	exponential quantity of infinite sets
[as a whole] in mass task:	the set consists of one element	infinite quantity of infinite sets

DISTINCTION BETWEEN SET MEMBERS in such set:

task: / subset:	task OAP	task TSP
in each individual case	the set consists of one element	in general case members of this set differ from each other
in each group case:	× × × × × ×	members of this set differ from each other
in each domain case:	× × × × × ×	members of this set differ from each other
[as a whole] in mass task:	the set consists of one element	members of this set differ from each other

DEPENDENCE FROM EACH OTHER of SET MEMBERS in such set:

task: / subset:	task OAP	task TSP
in each individual case	the set consists of one element	elements of such set never depend from each other
in each group case:	× × × × × ×	elements of such set never depend from each other
in each domain case:	× × × × × ×	elements of such set never depend from each other
[as a whole] in mass task:	the set consists of one element	elements of such set never depend from each other

Table 8: Comparison of fundamental properties of various subsets of individual cases of mass task OAP and mass task TSP

$L_b = fv_b(E_b) = fv_b(E_{b1}, E_{b2}, ..., E_{ba}, ...) = \{fv_{b1}(E_{b1}), fv_{b2}(E_{b2}), ..., fv_{ba}(E_{ba}), ...\}$ $a = 1, 2, 3, ...$
$|fv_b(E_b)| = |\{fv_{b1}(E_{b1}), fv_{b2}(E_{b2}), ..., fv_{ba}(E_{ba}), ...\}| = +\infty$

Theorem 8: In each domain case of TSP goal function of this case is not repeating in other domain cases infinite set of different from each other and independed on each other functions.

$L_c = fu_c(E_c) = fu_c(E_{c1}, E_{c2}, ..., E_{ca}, ...) = \{fu_{c1}(E_{c1}), fu_{c2}(E_{c2}), ..., fu_{ca}(E_{ca}), ...\}$ $a = 1, 2, 3, ...$
$|fu_c(E_c)| = |\{fu_{c1}(E_{c1}), fu_{c2}(E_{c2}), ..., fu_{ca}(E_{ca}), ...\}| = +\infty$

Theorem 9: In the mass task TSP goal function of this task is not repeating in other mass tasks infinite set of different from each other and independed on each other functions.

$L = f_0(E) = \{fu_1(E), fu_2(E), fu_3(E), ..., fu_c(E), ...\}$
$|f_0(E)| = |\{fu_1(E), fu_2(E), fu_3(E), ...\}| = +8$

Complexity of Algorithm of the Task TSP: Simple proofs of theorems 10 ... 17 are based that complexity [of solution] of a task are directly proportional to the following values:

- The quantity of obligatory parameter's values, which should process algorithm;
- The quantity of obligatory operations which must make algorithm.

Theorem 10: Algorithm's complexity of exact solution of individual case of the task TSP is an exponential function from the quantity of graph nodes in this individual case.

Theorem 11: Any individual algorithm of TSP is unique finite set of data and descriptions of operations over this data the implementation of which lead to the getting exact answer only of this individual task.

Theorem 12: Complexity of algorithm of the exact solution of any group case of TSP is equal to infinity.

Theorem 13: Any group algorithm of TSP is unique infinite set of unique individual algorithms.

Theorem 14: Complexity of algorithm of the exact solution of any domain case of TSP is equal to infinity.

Theorem 15: Any domain algorithm of TSP is unique infinite set of unique individual algorithms.

Theorem 16: Complexity of algorithm of the exact solution of the mass task TSP is equal to infinity.

The most effective algorithm for the task TSP is exponential exhaustive search.

Effective Algorithms for Other NP-Complete Tasks: Theorem 18: There is no effective (including any polynomial) algorithm of guaranteed reception in general case the exact answer for one NP-complete task, $P \neq NP$

Proof:

- All NP-complete tasks polynomially reduces to the task TSP.
- $FPA_{TSP} \neq FPR_{TSP}$, $ACr_{TSP} < 1$
- Uncertainty that exists in the task TSP can only be removed by adding new members to FPA_{TSP}.
- Let's suppose that for a certain task T_{mg} possible to create an efficient algorithm.

$FPA_{Tmg} = FPR_{Tmg}$, $ACr_{Tmg} = 1$

- TSP can not be reduced to T_{mg}: it's impossible polynomially (or otherwise) to convert the member that is not in the set FPA_{TSP} into member that is in the set FPA_{Tmg} (Fig. 8).
- If T_{mg} can not be reduce to a NP-complete task TSP, that means that T_{mg} is not NP-complete task.
- Consequently, NP-complete tasks for which it is possible to create effective algorithm, don't exist.
- Other tasks, that belong to the class NP and are not the task of the class P, polynomially reduced to TSP.
- Consequently, for this tasks holds everything said here about NP-complete tasks.
- For the tasks of the class P there exist polynomial exact algorithms.
- It means, that the class NP does not coincide with the class P: $P \neq NP$.

Comment: Using of concept "availability" give the obvious illustrations of affinity of problems "P vs NP" and "primes".

In Closing: Irregularity is the lack of a uniform law (or, equivalently, a law that is the set of many independent from each other laws).

Accident is law inaccessible to the detached onlooker.

The chaos is an irregular accident.

Essence of the problem P vs NP: it is inalienable chaos in the discrete sets with exponential (i.e. as much as big) quantity of set members.

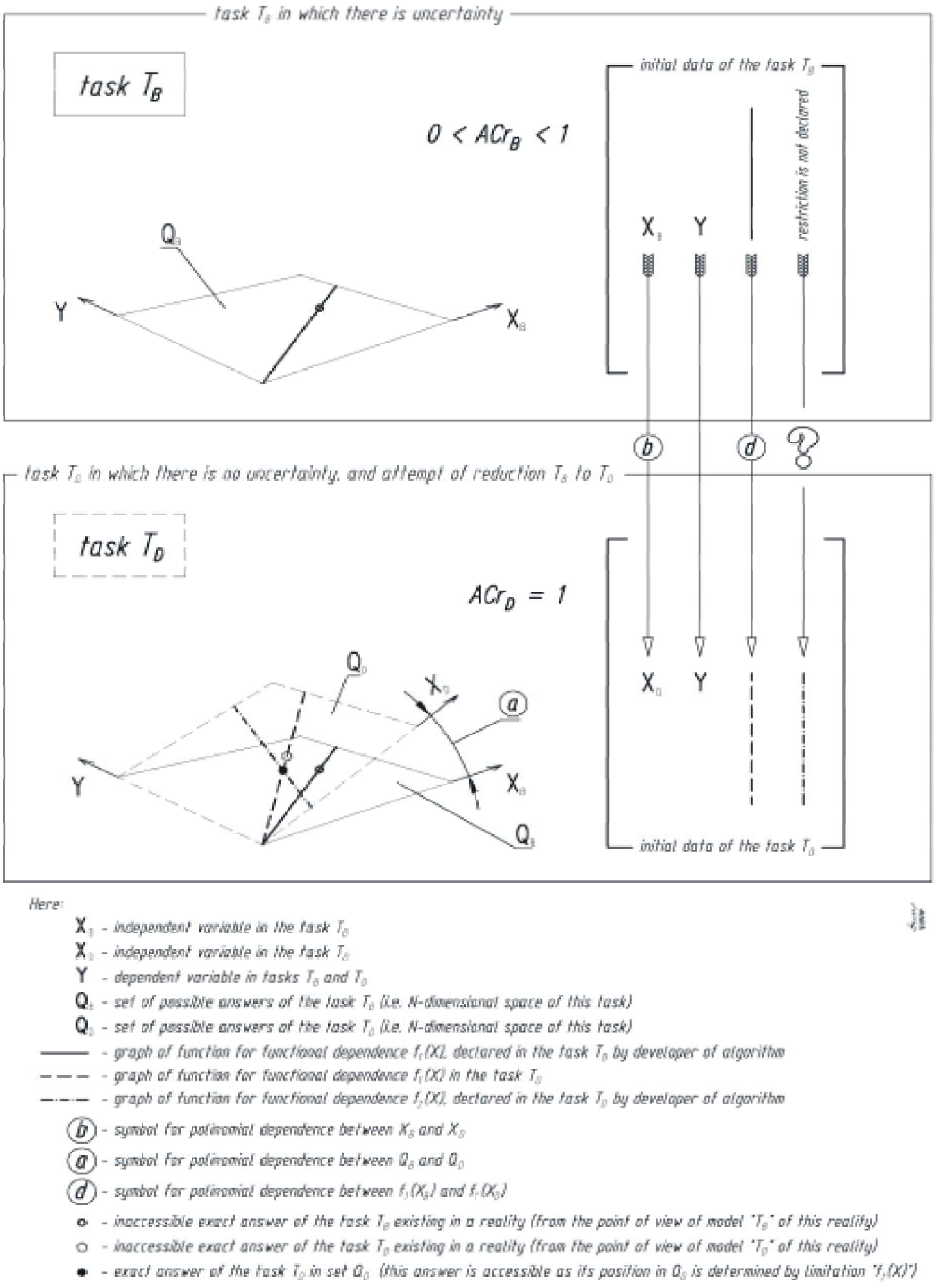

task T_B in which there is uncertainty
task T_B
0 < ACr_B < 1
initial data of the task T_B
restriction is not declared
X_B
Y
Q_B
b
d
?
task T_D in which there is no uncertainty, and attempt of reduction T_B to T_D
task T_D
ACr_D = 1
Q_D
X_D
a
Q_B
initial data of the task T_D
Here:
X_B - independent variable in the task T_B
X_D - independent variable in the task T_D
Y - dependent variable in tasks T_B and T_D
Q_B - set of possible answers of the task T_B (i.e. N-dimensional space of this task)
Q_D - set of possible answers of the task T_D (i.e. N-dimensional space of this task)
- graph of function for functional dependence f_1(X), declared in the task T_B by developer of algorithm
- graph of function for functional dependence f_1(X) in the task T_D
- graph of function for functional dependence f_2(X), declared in the task T_D by developer of algorithm
b - symbol for polinomial dependence between X_B and X_D
a - symbol for polinomial dependence between Q_B and Q_D
d - symbol for polinomial dependence between f_1(X_B) and f_1(X_D)
- inaccessible exact answer of the task T_B existing in a reality (from the point of view of model "T_B" of this reality)
- inaccessible exact answer of the task T_D existing in a reality (from the point of view of model "T_D" of this reality)
- exact answer of the task T_D in set Q_D (this answer is accessible as its position in Q_D is determined by limitation "f_2(X)")

Fig. 8: The simplified graphic interpretation of polinomial reduction of one task to anather

CONCLUSIONS

- P≠NP
- “Concierge’s key” is not existing.
- There is a much more interesting problem – “P[ersonal] C[omputer] vs NP”.
- The reason of fact “P≠NP” is fundamental and in general case this reason cannot be removed irrespective of a level of development of a science and technologies.

REFERENCES

1. Cook, S., 1971. The complexity of theorem-proving procedures. Proceeding of ASM STOC’, 71: 151-158.
2. Cook, S., 2006. The P versus NP Problem. Date Views 18.10.2006 http://www.claymath.org/millennium/P_vs_NP/Official_Problem_Description.pdf.
3. Garey, M.R. and D.S. Johnson, 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman, pp: 340.
4. Barrón-Romero, C., 2010. The Complexity Of The NP-Class. Date Views 11.06.2010 www.arXiv.org > cs > arXiv: 1006.2218.
5. Deolalikar, V., 2010. P≠NP. Date Views 06.08.2010 preprint.
6. Kobayashi, K., 2012. Topological approach to solve P versus NP. Date Views 06.02.2012 www.arXiv.org > cs > arXiv: 1202.1194.
7. Valeyev, R., 2010. P vs NP, PC vs NP or why it’s impossible to solve precisely the task about a traveling salesman. Date Views 21.05.2011 http://www.rst-valeyev.ru/page.php?id=spec.
8. Valeyev, R., 2012. Nontraditional algorithms for logistical tasks. Ostrovityanin, pp: 160.