Study of the NP-completeness of the compact table problem

Jean-Christophe Dubacq, Jean-Yves Moyen

Journées Automates Cellulaires 2008 (Uzès), pp. 228-237

STUDY OF THE NP-COMPLETENESS OF THE COMPACT TABLE PROBLEM

JEAN-CHRISTOPHE DUBACQ [1] AND JEAN-YVES MOYEN [1]

[1] LIPN/UMR 7030, CNRS/Université Paris XIII — 93430 Villetaneuse, France
URL: http://www-lipn.univ-paris13.fr

Abstract. The problem of compact tables is to maximise the overlap when building a word that is to include permutations of every given words (all the words being the same length). This problem is shown to be NP-complete in the general case, and some specific restrictions are studied.

1. Presentation of the problem

1.1. Informal description and examples

In several fields, it can be helpful to give a compact representation of a two dimensional table. We are interested in the case of random output tables, where the table is used to express the possible outcomes in n different initial conditions, where results can be described qualitatively with several results. These results can be represented several times for a given initial condition (to express discrete probabilities). We shall suppose that the number of possible outcomes (with multiple occurrences counted multiple times) is always the same (it could be 100 for a simplified percentage scheme, for example). This common number is called the amplitude of the table (ℓ).

For example, one can have the following table (on the left) where each row corresponds to a different set of initial conditions and each column corresponds to a different result. The numbers correspond to the possible outcomes, *i.e.* in case A, outcome α happens 30% of the time, outcome β happens 40% of the time and outcome γ happens with a probability of 30%. The same table can also be explicitly expanded (on the right).

	α	β	γ	δ
A	30%	40%	30%	0%
B	10%	30%	20%	40%
C	10%	0%	60%	30%
D	20%	20%	40%	20%

	1	2	3	4	5	6	7	8	9	10
A	α	α	α	β	β	β	β	γ	γ	γ
B	α	β	β	β	γ	γ	δ	δ	δ	δ
C	α	γ	γ	γ	γ	γ	γ	δ	δ	δ
D	α	α	β	β	γ	γ	γ	γ	δ	δ

Now, to find the outcome in, *e.g.* case B, one can computes a random number between 1 and 10 and directly look into the table for the corresponding result in the 'B' line.

2000 ACM Subject Classification: 68Q17, 68Q45, 91A90.
Key words and phrases: Probabilistic automaton, NP-completeness, Gaming theory.

If some of these results are common to several initial conditions, then it is possible to give a compact representation of the table. Instead of a $n \times \ell$ bi-dimensional table, we could express each initial condition as an offset in a one-dimensional table, that would cover all possible cases. It is always possible to do such a transformation as follows: give an offset of $i \times \ell$ to initial condition number i, and simply put all possible outcomes in a single table (in the order of the initial conditions). Initial condition 0 will use the 0 to $\ell - 1$ first possible outputs, initial condition 1 will use the ℓ to $2\ell - 1$ outputs, etc. It is quite obvious that the two formulations are equivalent.

The previous table can be "linearised" as follows :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
α	α	α	β	β	β	β	γ	γ	γ	α	β	β	β	γ	γ	δ	δ	δ	δ
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
α	γ	γ	γ	γ	γ	γ	δ	δ	δ	α	α	β	β	γ	γ	γ	γ	δ	δ

Then the result, *e.g* for case C can be directly read in the table by computing a random number between 21 and 30, or, rather, by computing a random number between 1 and 10 and adding an offset of 20.

Obviously, this linearised table is as big as the previous explicit table (40 results) and nothing have been gained that way.

The order of the initial conditions, however, is not important to the user of the table. The order of the outcomes is also not important. Therefore, one could move around initial conditions and outcomes so that some overlapping between results may appear.

Consider the following table :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
δ	δ	δ	δ	γ	γ	α	β	β	β	β	α	α	γ	γ	γ	γ	δ	δ	δ	γ	γ

Now, the result in each case can be read by computing a random number between 1 and 10 and adding an offset equal to:
A: +6, B: +0, C: +12, D: +9.

The resulting table is much smaller than the previous one (22 boxes instead of 40).

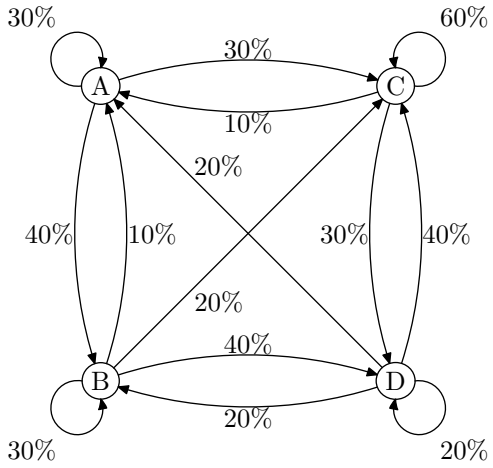
This kind of tables is, among other, used in simulation processes with simple models, such as the one you can find in *e.g.* wargames (historical simulation games). In wargames, a single action (such as a battle between armies) is usually resolved by rolling a die and looking for the result in a table. In this case, the initial conditions of the action (such as the relative strength of each army) determine the specific case and several outcomes (such as winning or loosing the battle with more or less casualties) can happen. To make things as unambiguous as possible between all players, it is better to have an explicit table rather than simply listing the probabilities of each outcome (which would be perfect in a computer-driven simulation but can lead to unnecessarily complicated computations as well as arguments over interpretation of the result in case of humans actually rolling dice). Obviously, compacting the tables allows to gain space and can thus be of great interest. Multi-dimensional tables are also very complex to remember (when several characteristics such as terrain, weather, strengths, etc. come into play in a single outcome) and linearised tables add to the playability in such games.

This is this kind of compact tables that this article studies (*e.g.* the assault table in [3]).

If the number of cases is less than or equal to the number of possible outcomes, this problem can be seen as the way of compacting transition tables for probabilistic automata. A probabilistic automaton is a normal (usually non-deterministic) automaton with weights

on the various transitions; when a transition is made from some state, the probability a given transition will occur is given by the normalised weights of all possible transitions starting from the current state. If the normalised weights can be expressed as fractions with a common denominator, then the transition table of the probabilistic automaton can be expressed with partial overlaps as described above. In this case, the set of results and the set of initial conditions will be the same, namely the set of possible states of the automaton. For example, one can consider the previous table to be the transition table of a probabilistic automaton with 4 states. In state D, the automaton can goes to state A (result α) with probability 20%, in state B (result β) with probability 20% and so on.

The whole automaton can be depicted as follows:



The problem can also be seen as finding the shortest word containing a permutation of each of the words given as input. With our running example, there will be 4 words: $A = \alpha\alpha\alpha\beta\beta\beta\beta\gamma\gamma\gamma$, $B = \alpha\beta\beta\beta\gamma\gamma\delta\delta\delta\delta$, $C = \alpha\gamma\gamma\gamma\gamma\gamma\gamma\delta\delta\delta$ and $D = \alpha\alpha\beta\beta\gamma\gamma\gamma\gamma\delta\delta$ and there exists a word of length 22 containing a permutation of each of these words: $\delta\delta\delta\delta\gamma\gamma\alpha\beta\beta\beta\beta\alpha\alpha\gamma\gamma\gamma\gamma\delta\delta\delta\gamma\gamma$. Of course, the letters in each of the words (A, B, C, D) do not need to be ordered in each instance of the problem.

This leads to other applications to this problem: giving the shortest possible string that can contain permutations of a set of given strings may be of interest in the field of biology. A DNA molecule (or a strand of proteins) could be replicated several times, and separated in many sequences by a physical process. With simple weight analysis techniques, the composition (with no knowledge of the order of the components) of each string could be determined. From there, the shortest (and thus most likely) original string can be computed by looking for possible overlaps in the outcomes.

For example, consider a DNA molecule whose composition is unknown. Since the four bases have different weights[1], the weight of the molecule gives indication on its composition (especially if the molecule is small). However, this is not sufficient to precisely determine the molecule (that is, the order in which the bases appear in it).

By well known physical methods, it is possible to replicate the molecule several times. Then, each copy can be split into smaller parts. The parts can be sorted by weight using centrifugation. Hopefully, the parts will be small enough so that the exact composition of each part can be deducted from its weight (otherwise, one need to cut them again).

[1]Adenine weights $135.127 g/mol$, Thymine weights $126.113 g/mol$, Cytosine weights $111.300 g/mol$ and Guanine weights $151.126 g/mol$.

Now, we have a set of words (the parts, whose composition is known) and we want to find a larger word (the whole molecule) such that it contains a permutation of each of the original words. The shortest solution is the more likely because a longer solution would probably have generated more different parts[2]. This is exactly the compact table problem.

1.2. Formal definition

The formal definition of the compact table problem is as follows:

Function Problem 1 (Compact table)**.**

Instance: *An alphabet Σ, an integer ℓ, a set of words $S \subset \Sigma^\ell$*
Answer: *The minimal length k of a word $\tau \in \Sigma^k$ such that for any word $u \in S$, there exists a permutation σ and words v and w such that $\tau = v \cdot \sigma(u) \cdot w$.*

This problem is naturally associated to a decision problem, which we will show to be NP-complete in the general case.

Decision Problem 1 (Compact table)**.**

Instance: *An alphabet Σ, an integer ℓ, a set of words $S \subset \Sigma^\ell$, an integer k*
Answer: YES *if there exists a word $\tau \in \Sigma^k$ such that for any word $u \in S$, there exists a permutation σ and words v and w such that $\tau = v \cdot \sigma(u) \cdot w$,* NO *in all other cases.*

2. General Case

To show that COMPACT TABLE is NP-complete, we shall first show that it is in the NP complexity class, and then that any instance of HAMILTONIAN PATH can be transformed in an instance of COMPACT TABLE, such that the answer to the two problems is the same (HAMILTONIAN PATH is a well-known NP-complete problem, see [10, 8]).

Decision Problem 2 (Hamiltonian path)**.**

Instance: *A graph $G = (V, E)$ ($n = |V|$)*
Answer: YES *if there exists a path $(v_1, e_1, v_2, e_2 \ldots, e_{n-2}, v_{n-1}, e_{n-1}, v_n)$ (such that $e_i = (v_i, v_{i+1})$) passing through all vertices of G,* NO *if not.*

COMPACT TABLE is in NP:

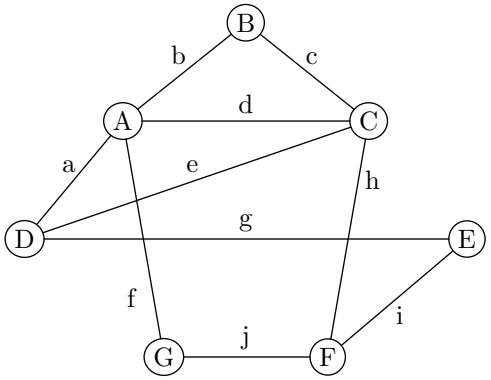
- The size of the inputs is no smaller than $|S| \times \ell$ because there are $|S|$ words of length ℓ each. The size of the result, τ, is smaller than $|S| \times \ell$ because there always exists a solution of length $|S| \times \ell$ (as shown in the examples above).
- Given τ (guessed non-deterministically), one first computes all the sub-words of τ of length ℓ (there are $|\tau| - \ell + 1$ such words) and compare each of them to each of the words in S (leading to $(|\tau| - \ell + 1) \times |S|$ comparisons between words).
- Each comparison between words can be done by first sorting the letters of the two words and then comparing the sorted words letter by letter. This require $O(\ell \log(\ell))$ comparisons for sorting and ℓ comparisons afterwards.
- Hence, the total number of comparisons (between letters) is $O(((|\tau| - \ell + 1) \times |S|) \times (\ell \log(\ell))) = O(|S|^2 \times \ell^2 \times \log(\ell))$.

[2]This means that the parts must not be too short, otherwise the same part may come from different places of the original molecule. However, this can be detected while weighting: if one part appear twice as many time as each other, then this is probably two identical parts.

Let us describe now how we shall transform an instance of Hamiltonian Path in an instance of Compact Table. Let $G = (V, E)$ be a graph, ℓ be the maximum degree of G and $n = |V|$ be the number of vertices in G.

Construction: We define Σ to be the set $E \cup V$. Each vertex v is associated to a word τ_v of Σ^ℓ which is the set of edges adjacent to v (in no particular order) and completed (since G is not forced to be regular) by as many occurrences of v as deemed necessary. k is determined to be $n(\ell - 1) + 1$.

The following graph (on the left) is thus transformed into the words on the right:



- $\tau_A = abdf$
- $\tau_B = BBbc$
- $\tau_C = cdeh$
- $\tau_D = Daeg$
- $\tau_E = EEgi$
- $\tau_F = Fhij$
- $\tau_G = GGjf$

This graph admits a Hamiltonian path $ABCDEFG$ corresponding to the word (of length $n(\ell - 1) + 1 = 22$) $\tau = adf\mathbf{b}BB\mathbf{c}dh\mathbf{e}Da\mathbf{g}EE\mathbf{i}Fh\mathbf{j}GGf$; we build τ by choosing overlapping letters (shown in bold) corresponding to the edge linking consecutive vertices.

If τ exists and satisfies all conditions. We have to show that G admits a Hamiltonian path. Given the definition of the set S, the only way two words τ_v and $\tau_{v'}$ may overlap (even with permutations) is if v and v' share an edge (all other symbols are distinct). They can, at this point, overlap by one letter (edge (v, v')). The only way the final string τ can be of length $n(\ell-1)+1$ is if there are $n-1$ overlaps (all words have to be present, and this is a total of $n\ell$ letters). If this is the case, one can find a sequence of edges that join vertices and thus a sequence $(v_0, e_0, \ldots, e_{n-2}, v_{n-1})$ of adjacent vertices and edges. This path passes through all vertices exactly once: if the string were redundant (one has not n but $m > n$ vertices), its length would be $m\ell - m + 1$. This is larger than k for all $m > n$. Thus, if τ exists, there exists a Hamiltonian path in G.

If G admits a Hamiltonian path. We have to show that there exists a word τ of length $n(\ell - 1) + 1$ that contains at least some permutation of the word associated to any vertex v of G. Consider one of the Hamiltonian paths $(v_1, e_1, v_2, e_2, \ldots, v_{n-1}, e_{n-1}, v_n)$ and define τ as follows:

$$\begin{cases} \tau &= u_1 \cdot e_1 \cdot u_2 \cdot e_2 \cdot u_3 \cdot \ldots \cdot u_n \\ u_i &= \tau_{v_i} \text{ with } e_i \text{ and } e_{i-1} \text{ removed, } (1 < i < n) \\ u_1 &= \tau_{v_1} \text{ with } e_1 \text{ removed} \\ u_n &= \tau_{v_n} \text{ with } e_{n-1} \text{ removed} \end{cases}$$

It is quite straightforward that τ is of length $(n-2)(\ell-2) + 2(\ell-1) + (n-1)$, i.e. $n(\ell-2) + 2 + n - 1 = n(\ell-1) + 1$ and contains a permutation of τ_v for any vertex v (just write τ_v with the edge preceding v in the Hamiltonian path first and the edge following v in the Hamiltonian path last).

#P-completeness. We shall not introduce the counting problems versions of all these problems, but we mention the result here for the sake of completeness. The counting problem associated to HAMILTONIAN PATH is a #P-complete problem, as shown for example by theorem 18.2 of [13]. However, our transformation does not preserve the number of solutions (it is not *parcimonious*), and as such is not usable to determine whether COMPACT TABLE is a #P-complete problem. This is because of the allowed permutations: in our example above, the word $\tau' = daf\mathbf{b}BB\mathbf{c}dh\mathbf{e}Da\mathbf{g}EE\mathbf{i}Fh\mathbf{j}GGf$ (remark the difference in the beginning of the word) is another solution of the COMPACT TABLE instance, but is in correspondance to the exact same solution of HAMILTONIAN PATH. The number of solutions of COMPACT TABLE matching one precise HAMILTONIAN PATH solution is easy to compute (and they do not overlap), it is $(\ell-1)^2 \prod_{1\leq i\leq n} \frac{(\ell-2)!}{(\ell-d(i))!}$ (there are $\frac{(\ell-2)!}{(\ell-d(i))!}$ allowed permutations for the part of the word corresponding to each vertex, and $(\ell-1)$ times more for the initial/ending vertices). This number does however depend only on the chosen initial instance of HAMILTONIAN PATH, which means that COMPACT TABLE is #P-hard (if one can count the number of solutions of any instance of COMPACT TABLE, one can count the number of solutions of any instance of HAMILTONIAN PATH).

3. The fixed-amplitude case

3.1. Amplitude larger than 3

Let us consider the following family of decision problems (indexed by ℓ):

Decision Problem 3 (Compact table of order ℓ)**.**

Instance: *An alphabet Σ, a set of words $S \subset \Sigma^\ell$, an integer k*

Answer: YES *if there exists a word $\tau \in \Sigma^k$ such that for any word $u \in S$, there exists a permutation σ and words v and w such that $\tau = v \cdot \sigma(u) \cdot w$,* NO *in all other cases.*

It is obvious that the case $\ell = 1$ is polynomial. We can show that for any $\ell > 2$, COMPACT TABLE OF ORDER ℓ is still NP-complete, since the following family of problems is also NP-complete for all $d > 2$ (see [9, 12]):

Decision Problem 4 (Hamiltonian path in d-bounded degree graphs)**.**

Instance: *A graph $G = (V, E)$ ($n = |G|$) of maximal degree d*

Answer: YES *if there exists a path $(v_0, e_0, \ldots, e_{n-2}, v_{n-1})$ (such that $e_i = (v_i, v_{i+1})$) passing through all vertices of G,* NO *if not.*

The very same construction we used shows that for any $\ell > 2$, the COMPACT TABLE OF ORDER ℓ remains NP-complete.

3.2. Amplitude 2

We prove that in the case of amplitude 2, the problem becomes polynomial. In this case, each initial condition leads to an alternative between two results. We will reduce the problem to the following graph problem:

Function Problem 2 (Eulerian path)**.**

Instance: *An undirected $G = (V, E)$*

Answer: *A minimum set E' such that $G' = (V, E \cup E')$ is Eulerian.*

Let us consider an instance of Compact Table of order 2: let Σ be an alphabet and $S \subset \Sigma^2$ be a set of words[3] of length 2. Let $n = |S|$ be the number of words and $m = |\Sigma|$ be the number of letters of the alphabet.

We build a graph G with m vertices and n edges in the following way:

- There is a vertex α for each letter of the alphabet $\alpha \in \Sigma$.
- There is an edge between α and β if and only if the word $\alpha\beta$ is in S.

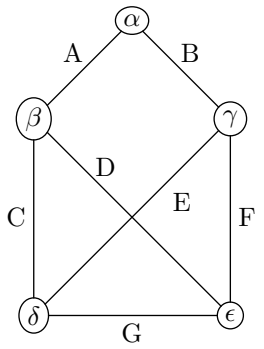
We shall now prove that the instance of Compact Table of order 2 admits a solution of length $n + k + 1$ if and only if G can be made Eulerian by adding k edges.

First, let G' be an Eulerian graph obtained by adding k edges to G. Consider an Eulerian path in G'. By enumerating the vertices traversed by the path, we obtain a word in Σ^* of length $n+k+1$ (there are $n+k$ edges, hence $n+k+1$ vertices). This word contains a permutation of each of the words in S because each of these words correspond to an edge in G, hence an edge in G', and the path has to go through all edges of G' by definition of Eulerian paths. Hence, if G can be made Eulerian by adding k edges to it, then Compact Table of order 2 admits a solution of length $n + k + 1$.

Conversely, if Compact Table of order 2 admits a solution τ of length $n + k + 1$. Let G'' be the smallest complete graph containing G. τ is the description of a path in G'' going through $n + k$ edges. Since τ contains a permutation of every word in S, all the edges of G belong to this path. Let G' be the graph obtained from G'' by keeping only the edges along this path. By construction, G' is Eulerian and contains all the edges in G plus k additional edges. Hence, if Compact Table of order 2 admits a solution of length $n + k + 1$, G can be made Eulerian by adding k edges to it.

For example, the table on the left (in each case, both results have the same probability) leads to the graph on the right:

	α	β	γ	δ	ϵ
A	1	1			
B	1		1		
C		1		1	
D		1			1
E			1	1	
F			1		1
G				1	1



Here, the graph is not Eulerian but can be made so by adding a single edge, H, between β and γ. This leads, among other, to the Eulerian path $EHABFGCD$ corresponding to the word $\delta\gamma\beta\alpha\gamma\epsilon\delta\beta\epsilon$ of length 9 with the offsets:

A: +2, B: +3, C: +6, D: +7, E: +0, F: +4, G: +5.

Let G be a graph. The minimum number of edges ones need to add to make it Eulerian[4] can be computed in polynomial time. Let us describe graphs as having c connected components and $2n$ vertices of odd degree. We separate the connected components between those with vertices of odd degree (A, $a = |A|$) and the others (B, $b = |B|$). A graph is

[3]We consider here that the words in S are all different. Identical words can be dealt with by considering multigraphs instead of graphs.

[4]The resulting graph may in fact be a multigraph, but this does not matter for our problem

Eulerian when (a, b, n) is $(1, 0, 1)$ or $(0, 1, 0)$. We shall suppose that we are not in one of those two cases.

If one edge is added to G, these are all the possible moves (we consider the variation of (a, b, n) as moves in a three-dimensional space):

- α: Adding one edge going from one component to itself: either $[0, 0, 1]$ between two even vertices, $[0, 0, 0]$ between an even vertex and an odd vertex, $[0, 0, -1]$ between two odd vertices. There is a special case for the last one: the move could also be $[-1, 1, -1]$.
- β: Adding one edge between two components of A: $[-1, 0, 1]$ between two even vertices, $[-1, 0, 0]$ between an even vertex and an odd vertex, $[-1, 0, -1]$ between two odd vertices.
- γ: Adding one edge between one component of A and one of B: $[0, -1, 1]$ if the vertex in the component in A was of even degree, $[0, -1, 0]$ otherwise. There is always an even number of odd-degree vertices in a component, so a never decreases this way.
- δ: Adding one edge between two components of B: $[1, -2, 1]$ (always).

The number of edges to be added (if not zero), is at least $b + n - 1$ (no move decreases $b + n$ by more than one, and the goal is of value 1). We exhibit a greedy (polynomial) algorithm that adds exactly $b + n - 1$ edges, and is therefore optimal.

- If $a = 0$, then $n = 0$ and $b > 1$. The transformation $\delta\gamma^{b-2}$ leads us to the final state and is of length $b + n - 1 = b - 1$.
- If $a > 0$, then transformation $\beta^{a-1}\gamma^b\alpha^{n-a}$ leads us to the final state and is of length $b + n - 1$.
- In each case, there is only one subcase that decreases $b + n$; there may be some choice for the exact edge to be added.

This is the smallest number of edges one must add to make the graph Eulerian as shown by Fleury's algorithm (see [4, 5, 6]). Remark that this is related to the Chinese postman problem [11] and the rural postman problem.

4. Limited number of results

4.1. The 2-results case

We would like to point out that the Compact Table of order ℓ problem with a limited number of possible results becomes trivial (solvable in constant time), because with a limited number of results and a fixed amplitude, the number of different words is finite.

Even the general Compact Table problem is polynomial in the case of only two possible results, namely success and failure (compare this to a toss of coin, with different probabilities of win according to some initial conditions). One can use a sequence of m_1 times the first result "0" followed by m_2 times the second result "1", where m_1 is the largest number of "0" for any initial condition and m_2 is the largest number of "1". Each initial condition can be associated to an offset that sets the number of possible "0" and "1". This word is the shortest possible since any satisfying word must have at least m_1 "0" and m_2 "1". For example, if the amplitude is 4 and the words are: $0000, 1101, 1001$ then the word 0000111 is the shortest one admitting permutations of 0000, 0111 and 0011 as continuous sub-words (with offsets 0, 3 and 2 respectively).

However, it should be remarked that giving a whole table for such a problem is not the best way to implement it. Since there are only two results, it is sufficient to give the amplitude and the offsets which correspond to the probabilities of "1", all other cases being "0". This means that the problem remains the same if, instead of words, one is simply given the number of "0" and "1" in each case (a more compact representation than then "unary" one used above).

This 2-results case is also of practical use. In genetic epidemiology, when introducing a new method to detect which genes can cause a disease, one has to test the method. This test is usually done on a simulated population. Each member of it is generated from a pre-existing pool of genotypes, respecting actual ratio observed in the population. Each member is then assigned a disease status (either affected or not). The probability of being affected depends on the genotype (corresponding to pre-existing observations made on the same disease, if the test validate the method, then it can later be carried over to not yet studied diseases) [2]. This corresponds exactly to the 2-results case: there are only two possible outcomes (affected or not) and the probability of each outcome depends on the input (the genotype).

4.2. With 3 or more results

For the case of a restricted alphabet (3 or more), the question is still open (about whether the problem remains NP-complete). However, the first remark still stands: if enough distinct words are given, all possible outputs finally appear. The number of such words (with k being the number of results and ℓ being the amplitude of the words) is $\binom{\ell+k-1}{\ell}$. A simple proof of this: a word is given by the number of occurrences of each result, including 0. This is in bijection to the words on alphabet $\{x, y\}$ of length $\ell + k - 1$, in which one chooses $k - 1$ delimiters "y" separating runs of "x" (a run can be of length 0). The distance between any two delimiters is the number of occurrences of the corresponding result (the first result at the beginning of the word, followed by the second, etc.).

However, the question of whether there exists a word that contains all possible combinations of length $\binom{\ell+k-1}{\ell} + \ell - 1$ (which would be the minimal length of such a word) is still open, and is too complex to fit in this paper. The authors conjecture that it is at least possible to do it for three results.

5. Conclusion and Future Works

We would like to remark that the similar albeit different problem where the initial words are given but no permutation is allowed to find them in the final word τ is also NP-complete in the general case.

Decision Problem 5 (Compact ordered table)**.**

Instance: *An alphabet Σ, an integer ℓ, a set of words $S \subset \Sigma^\ell$, an integer k*

Answer: YES *if there exists a word $\tau \in \Sigma^k$ such that for any word $u \in S$, there exist words v and w such that $\tau = v \cdot u \cdot w$,* NO *in all other cases.*

The associated function problem was shown to be NP-hard in [7], and several approximation have been shown since (see e.g. [1]).

We have restricted ourselves here to words of the same length (ℓ). Obviously, if the input words can be of any length the problem is more general, hence harder (the reduction

from Hamiltonian Path still works) and the problem is also NP-complete (the proof of being in NP is the same as the one we present).

The question of whether or not the original problem is approximable is open. The heuristics for the compact ordered table may apply, but they are probably not very efficient. The originality of this work is that the permutations allowed might have been an ease to solve the problem, but being able to choose the order of overlapping sequences does not break the NP-hardness of the problem.

6. Thanks

We would like to thank Pierre Borgnat, for showing us the use of overlapping tables in wargames tables. We would like also to thank the anonymous referees for pointing out glaring mistakes, small typos, spelling errors found in the first draft and the interrogation about #P-completeness.

References

[1] Chris Armen and Clifford Stein. A 2 2/3 superstring approximation algorithm. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 88, 1998.

[2] Claire Bardel, Vincent Danjean, J.-P. Hugot, P. Darlu, and E. Génin. On the use of haplotype phylogeny to detect disease susceptibility loci. *BMC Genet.*, 6(1):24, May 2005.

[3] Pierre Borgnat, Bertrand Asseray, Jean-Yves Moyen, and Jean-Christophe Dubacq. Europa Universalis 8. `http://bamgames.free.fr/Europa/EU8/`, 2008.

[4] Reinhard Diestel. *Graph Theory*. Springer-Verlag, August 2005.

[5] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiæ scientiarum Petropolitanæ*, 8:128–140, 1741. Translations available in English.

[6] Henri Fleury. Deux problèmes de géométrie de situation. *Journal de Mathématiques élémentaires*, pages 257–261, 1883.

[7] John K. Gallant, David Maier, and James A. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, (20):50–58, 1980.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability*. Freeman, 1979.

[9] Michael R. Garey, David S. Johnson, and Robert E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal of Computing*, 5(4):704–714, 1976.

[10] Richard Manning Karp. *Complexity of Computer Computations*, chapter Reducibility amoung Combinatorial Problems. Raymond E. Miller and James W. Thatcher, new york: plenum press edition, 1972.

[11] Mei-Ko Kwan. Graphic programming using odd or even points. *Chinese Math.*, (1):273–277, 1962.

[12] Maciej Liśkiewicz, Mitsunori Ogihara, and Seinosuke Toda. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. *Theoretical Computer Science*, 304(1-3):129–156, 2003.

[13] Christos H. Papadimitriou. *Computational Complexity*, chapter 18, pages 439–443. Addison Wesley Longam, 1995.