4-1-2007

Decomposable Aggregability in Population Genetics and Evolutionary Computations: Algorithms and Computational Complexity

Vladik Kreinovich
University of Texas at El Paso, vladik@utep.edu

Max Shpak
University of Texas at El Paso, mshpak@utep.edu

Comments:
UTEP-CS-06-41a.
Published in: Arpad Kelemen, Ajith Abraham, and Yulan Liang (Eds.), Computational Intelligence in Medical Informatics, Springer-Verlag, Berlin-Heidelberg, 2008, pp. 69-92.

Decomposable Aggregability in Population Genetics and Evolutionary Computations: Algorithms and Computational Complexity

Vladik Kreinovich and Max Shpak

University of Texas at El Paso, El Paso, TX 79968, USA vladik@utep.edu, mshpak@utep.edu

Summary. Many dynamical systems are *decomposably aggregable* in the sense that one can divide their (micro) variables $x_1, \ldots, x_n$ into several (k) non-overlapping blocks and find combinations $y_1, \ldots, y_k$ of variables from these blocks (*macrovariables*) whose dynamics depend only on the initial values of the macrovariables. For example, the state of a biological population can be described by listing the frequencies x_i of different genotypes i; in this example, the corresponding functions $f_i(x_1, \ldots, x_n)$ describe the effects of mutation, recombination, and natural selection in each generation.

Another example of a system where detecting aggregability is important is a one that describes the dynamics of an evolutionary algorithm – which is formally equivalent to models from population genetics.

For very large systems, finding such an aggregation is often the only way to perform a meaningful analysis of such systems. Since aggregation is important, researchers have been trying to find a general efficient algorithm for detecting aggregability.

In this chapter, we show that in general, detecting aggregability is NP-hard even for linear systems, and thus (unless P=NP), we can only hope to find efficient detection algorithms for specific classes of systems. Moreover, even detecting *approximate* aggregability is NP-hard.

We also show that in the linear case, once the blocks are known, it is possible to efficiently find appropriate linear combinations y_a.

1 What is Aggregability

Many systems in nature can be described as *dynamical systems*, in which the state of a system at each moment of time is characterized by the values of (finitely many) variables $x_1, \ldots, x_n$, and the change of the state over time is described by an equation $x_i' = f_i(x_1, \ldots, x_n)$, where

- for continuous-time systems, in which the time t can take any real value, x_i' is the first time derivative of x_i:

$$\frac{dx_i}{dt} = f_i(x_1, \ldots, x_n); \tag{1}$$

- for discrete-time systems, in which the time t can only take integer values, x_i' is the value of x_i at the next moment of time:

$$x_i(t+1) = f_i(x_1(t), \ldots, x_n(t)). \tag{2}$$

For example, the state of a biological population can be described by listing the frequencies x_i of different genotypes i; in this example, the corresponding functions $f_i(x_1, \ldots, x_n)$ describe the effects of mutation, recombination, and natural selection.

For natural systems, the number of variables is often very large. For example, for a system with g loci on a chromosome in which each of these genes can have two possible allelic states, there are $n = 2^g$ possible genotypes. For large g, dues to the large number of state variables, the corresponding dynamics is extremely difficult to analyze.

Many biological systems (and in many systems from other fields such as economics [22] and queuing theory [1] etc.) are *aggregable* in the sense that for appropriate functions $y_a = c_a(x_1, \ldots, x_n)$, $a = 1, \ldots, k$ ($k \ll n$, equations (1) or (2) lead to simpler equations

$$\frac{dy_a}{dt} = h_a(y_1, \ldots, y_k) \tag{3}$$

or, correspondingly,

$$y_a(t+1) = h_i(y_1(t), \ldots, y_k(t)) \tag{4}$$

for appropriate functions $h_1, \ldots, h_k$.

In other words, if the dynamics of x_i is described by the equation (1) or (2), then for the resulting dynamics of $y_a = c_a(x_1, \ldots, x_n)$, we have, correspondingly, the equations (3) or (4).

The corresponding combinations $y_a = c_a(x_1, \ldots, x_n)$ (where c_a are functions from real numbers to real numbers) are called *macrovariables*. If a system is aggregable, then the original variables $x_1, \ldots, x_n$ are also called *microvariables*.

Aggregability property has been actively studied; see, e.g., [1, 9, 10, 12, 17, 18, 20, 21, 22]. In most practical cases, the aggregation is *decomposable* in the sense that variables $x_1, \ldots, x_n$ can be divided into non-overlapping blocks $I_1 = \{i(1,1), \ldots, i(1,n_1)\}, \ldots, I_k = \{i(k,1), \ldots, i(k,n_k)\}$ ($\cup I_a = \{1, \ldots, n\}$ and $I_a \cap I_b = \emptyset$ for $a \neq b$) with $n_1, \ldots, n_k$ elements $\left(\sum_{a=1}^{k} n_a = n\right)$ so that each macrovariable y_a depends only on the variables x_i from the a-th block: $y_1 = c_1(x_{i(1,1)}, \ldots, x_{i(1,n_1)}), \ldots, y_k = c_k(x_{i(k,1)}, \ldots, x_{i(k,n_k)})$; see, e.g., [20].

In this case, macrovariables take the form $y_a = c_a(x_{i(a,1)}, \ldots, x_{i(a,n_a)})$, where $I_a = \{i(a,1), \ldots, i(a,n_a)\}$; if the dynamics of x_i is described

by the equation (1) or (2), then for the resulting dynamics of $y_a = c_a(x_{i(a,1)}, \ldots, x_{i(a,n_a)})$, we have, correspondingly, the equations (3) or (4).

Since most practical aggregable systems are decomposable, in this chapter, we will concentrate on decomposable aggregations. For readability, we will simply refer to decomposable and aggregable systems as *aggregable*.

Comment. In analyzing aggregability, we should take into consideration that perfect aggregability usually occurs only in idealized mathematical models. In many practical situations, we only have *approximate* aggregability, so that the aggregate dynamics (3) or (4) differs only marginally from the actual microdynamics of the macrovariables variables $y_a = c_a(x_{i(a,1)}, \ldots, x_{i(a,n_a)})$.

Note that many dynamical systems are only approximately aggregable during certain time intervals in their evolution, or over certain subspaces of their state space [10, 22].

2 Exact Aggregability: A Simple Example

If there are g genetic loci such that allelic substitutions at each locus have identical effects, then instead of $n = 2^g$ original variables $x_{0\ldots0}, x_{0\ldots1}, \ldots, x_{1\ldots1}$, we can consider $k = g + 1$ ($\ll 2^g$) macrovariables $y_1 = x_{0\ldots0}$ (the frequency of the original state), $y_2 = x_{0\ldots01} + x_{0\ldots010} + \ldots + x_{0\ldots010\ldots0} + \ldots + x_{10\ldots0}$ (the frequency of states with a single mutation), y_3 – overall frequency of states with 2 mutations, $\ldots$, $y_{g+1} = x_{1\ldots1}$. This example is described in detail in [19, 20, 23]; let us give a brief description of the simplest case.

In this scenario, we assume that reproduction is asexual, without crossing-over (recombination). We assume the simplest model of non-overlapping generations. In this case, we can select the discrete time variable t in such a way that at moment t, only members of t-th generation are present.

Since there is no crossing-over, the only possible changes in a genotype are caused by mutations. Since individual mutations are rare events, the probability of multiple mutations is negligible. Therefore, in the corresponding first order approximation, we do not consider multiple mutations. In this approximation, each genotype can mutate into g possible *mutational neighbors*, corresponding to the change in each of g cites. For example, a genotype $00\ldots0$ can evolve into one of g mutational neighbors $10\ldots0$, $010\ldots0$, $\ldots$, $00\ldots01$. In precise terms, a genotype g' is a mutational neighbor of g if they differ only in one locus, i.e., if the Hamming distance between them is equal to 1.

The per-locus mutation rate, i.e., the probability that a mutation occurs at a given locus, is assumed to be constant; we will denote this constant by μ.

We also make a similar assumption about the *fitness* w_i of the genotype i, i.e., its rate of replication (expected number of offspring of an individual). Specifically, we assume that this fitness w_i depends only on the number of 0s and 1s in the description of i. In evolutionary computation, such a fitness function is called a *function of unitation*; see, e.g., [16] and Chapter 6 of [15].

In population genetics, this fitness function is often called the *Eigen model* after Manfred Eigen's work on error thresholds and quasispecies (see, e.g., [5]).

For this fitness function, we will denote the number of ones in a genotype i by o_i; then, the number of 0s in this genotype is equal to $g - o_i$, and the fitness w_i is equal to $w_i = w(o_i)$.

Let us describe the discrete-time genotype frequency dynamics under these assumptions. It is assumed that at moment t, the (relative) frequency of i-th genotype is equal to $x_i(t)$. Let N be the total number of individuals; this means that out of N individuals, we have $N \cdot x_i(t)$ individuals of genotype i.

The replication rate w_i means that $N \cdot x_i(t)$ individuals of genotype i result, in the next generation, in $w_i \cdot (N \cdot x_i(t))$ individuals. Thus, the total number of individuals in the next generation $t+1$ is equal to $N(t+1) = \sum_i w_i \cdot (N \cdot x_i(t)) = N(t) \cdot \overline{w}$, where $\overline{w} = \sum_i w_i \cdot x_i(t)$. The ratio $\overline{w} = \dfrac{N(t+1)}{N(t)}$ is called the *mean fitness*.

An individual of the next generation has a genotype i if either its parent had the same genotype and did not mutate, or if its parent had a genotype j which differed from i in a single locus, and it mutated at this locus at which it mutated. The mutation rate at each locus is equal to μ, and we ignore multiple mutations. Thus, the probability that one of g loci mutates is $g \cdot \mu$, and the probability of no mutation is $1 - g \cdot \mu$. As a result, the total number of individuals N_i with genotype i in the next generation can be determined as

$$N_i(t+1) = (1 - g \cdot \mu) \cdot w_i \cdot x_i(t) + \sum_{j \sim i} \mu \cdot x_j(t), \tag{5}$$

where $j \sim i$ means that j and i are mutational neighbors (i.e., that their Hamming distance is 1). Dividing the above formula for the number of individuals $N_i(t+1)$ with genotype i by the expression for the total number of individuals $N(t+1)$, and canceling the common factor N in both numerator and denominator, we get the desired formula for the frequencies $x_i(t+1)$ at the next moment of time:

$$x_i(t+1) = \frac{1}{\overline{w}} \cdot \left[(1 - g \cdot \mu) \cdot w_i \cdot x_i + \sum_{j \sim i} \mu \cdot w_j \cdot x_j \right], \tag{6}$$

where $\overline{w} = \sum_i w_i \cdot x_i$ is the mean fitness.

In this example, it is natural to subdivide all the variables x_i into $g+1$ blocks $I_1, \ldots, I_{g+1}$, where I_a consists of all genotypes i with $o_i = a - 1$, i.e., with $a-1$ zeros, and, correspondingly, $g - (a-1)$ ones. Each genotype i has g neighbors corresponding to a change at one locus. Thus, each genotype i from the class I_a has $a-1$ mutational neighbors from the class I_{a+1} (with an additional "1") and $g - (a-1)$ mutational neighbors from the class I_{a-1} (with one fewer "1").

As macrovariables, we can take $y_a = \sum\limits_{i \in I_a} x_i$. Let us find the dynamic equations for these macrovariables. Each genotype i from the class I_a, by definition, has $a-1$ zeros and $g-(a-1)$ ones. Since we are only allowing single mutations, there are only three possibility to get this genotype in the next generation:

- One possibility is that we had exactly this genotype in the previous generation, and no mutation occurred. As we have mentioned, the probability that no mutations occurred is equal to $1 - g \cdot \mu$. Thus, the corresponding term in x_i' is $(1 - g \cdot \mu) \cdot x_i$. By adding these terms, we conclude that the corresponding term in $y_a' = \sum\limits_{i \in I_a} x_i'$ is equal to $(1 - g \cdot \mu) \cdot y_a$.
- Another possibility is that we had one more zero, i.e., the original genotype x_j belonged to the class I_{a+1}, and there was a mutation that changed an additional zero to a one. The probability of such a mutation is μ, so the corresponding term in $y_a' = \sum\limits_{i \in I_a} x_i'$ is thus proportional to $\mu \cdot \sum\limits_{j \in I_{a+1}} x_j$. Each genotype $j \in I_{a+1}$, with a zeros, has a possible mutations that lead to a genotype from I_a. Thus, in the formula for $y_a' = \sum\limits_{i \in I_a} x_i'$, we have a occurrences of each term x_j for $j \in I_{a+1}$. So, the corresponding term in y_a' takes the form $a \cdot \mu \cdot y_{a+1}$.
- Finally, it is possible that original genotype x_j belonged to the class I_{a-1}, with one more "1" than in x_j', and there was a mutation that changed one of its 1s to a zero. The probability of such a mutation is μ, so the corresponding term in $y_a' = \sum\limits_{i \in I_a} x_i'$ is thus proportional to $\mu \cdot \sum\limits_{j \in I_{a-1}} x_j$. Each genotype $j \in I_{a-1}$, with $g-a$ ones, has $g-a$ possible mutations that lead to a genotype from I_a. Thus, in the formula for $y_a' = \sum\limits_{i \in I_a} x_i'$, we have $g-a$ occurrences of each term x_j for $j \in I_{a-1}$. So, the corresponding term in y_a' takes the form $(g-a) \cdot \mu \cdot y_{a-1}$.

We assumed that the fitness w_i of a genotype $i \in I_a$ depends only on the number $a-1$ of zeros in this genotype, i.e., $w_i = w(a)$ for all $i \in i_a$. Thus, $\overline{w} = \sum\limits_i w_i \cdot x_i = \sum\limits_{a=1}^{g+1} w(a) \cdot y_a$, and the dynamics of the macrovariables y_a takes the form:

$$y_a(t+1) = \frac{1}{\overline{w}} \cdot \left[(1 - g \cdot \mu) \cdot y_a + a \cdot \mu \cdot y_{a+1} + (g-a) \cdot \mu \cdot y_{a-1}\right], \qquad (7)$$

where

$$\overline{w} \stackrel{\text{def}}{=} \sum_{a=1}^{g+1} w(a) \cdot y_a. \qquad (8)$$

3 Detecting Aggregability is Important

In many actual problems, the variables can be subdivided into blocks based on identity or symmetry properties. In the previous example from population genetics (or genetic algorithms), if we know the fitness effects of all the genes, it is natural to group together all the genotypes with similar effects.

Another example of a system where detecting aggregability is important is a one that describes the dynamics of an evolutionary algorithm – which is formally equivalent to models from population genetics. In genetic algorithms, the resulting group of "genes" is called a *schema* [8].

Genetic algorithms are used in bioinformatics for multiple sequence alignment and for a number of other standard bioinformatics applications; see, e.g., [13, 14]. Genetic algorithms are also used in predicting RNA and protein secondary structure.

Other examples of multivariable biological systems where aggregation of variables may prove useful include gene regulatory networks [3], metabolic control theory [6], and community ecology [9]. In all of these systems, there are always blocks of microvariables that behave in a similar or concerted fashion.

In many other situations, however, we only know the equations (1) or (2) (we may know these equation from the the analysis of the empirical data), but we do not yet know how to properly divide and combine the variables. For example, one may not know the functional role or epistatic interactions of genes on a chromosome a priori, only the phenotypes or Darwinian fitnesses of different genotypes. In such situations, it is important to be able to detect whether an aggregation is possible – and, if possible, to find such an aggregation. The aggregation itself may be instructive as to the function and interaction of genes, and may inform one as to which system components are relevant. For a detailed discussion see, e.g., [1, 9, 10, 12, 17, 18, 20, 21, 22]. (The reader should be aware that some of these papers deal with the general notion of aggregability, when blocks may overlap.)

Usually, we have some *partial* information about the variables – e.g., we may know that a certain variable x_i should affect one of the combinations y_a. In such situations, it is desirable to restrict the search to blocks which are consistent with this partial information.

4 What We Do in This Chapter

For some special systems with known symmetry properties, there exist efficient techniques that detect decomposable aggregability and find the corresponding aggregations. Since it is important to detect aggregability, researchers have been trying to find a *general* efficient method for its detection.

In this chapter, we show that even in the simplest case when the system is *linear* (i.e., all the dependencies f_i in (1), (2) are linear), the number of classes

is $k = 2$, and the additional information consists of a single variable that has to be involved in one of the combinations y_a, the problem of detecting decomposable aggregability is NP-hard. This means that even for linear systems (unless P=NP), there is no hope of finding a *general* method for detecting decomposable aggregability; we should therefore concentrate our efforts on detecting decomposable aggregability for *specific* classes of dynamic systems.

Comment. For readers who are not very familiar with the notion of NP-hardness, here is a brief and informal explanation.

The complexity of a computational problem is usually described by the computation time that is needed to solve this problem. This time grows with the number of variables (and with the number of bits which are needed to represent each of these variables).

For some computational problems, this time grows as a polynomial of the size n of the input. For example, the standard algorithms for multiplying two $n \times n$ matrices or solving a system of linear equations with n unknowns grows as $\text{const} \cdot n^3$. For large n, this is still feasible. These problems are called *polynomial time* (or P, for short), and the class of such problem is denoted by P.

For some other computational problems, however, the computation time grows exponentially with n, as 2^n or even faster. For example, this growth occurs when we need to search for a subset of the set of n variables, and we need to do an exhaustive search over all 2^n subsets in order to find the desired set. For such algorithms, for reasonable $n \approx 300 - 400$, the number of computational steps exceeds the number of particles in the Universe; thus, such exhaustive-search algorithms are not practically feasible.

In most of these problems, once we have guessed a solution, we can check, in feasible (polynomial) time, whether this guess is indeed a correct solution. Such problems can be "solved" in polynomial time on a hypothetical "non-deterministic" machine, i.e., on a Turing machine that allows non-deterministic (= guess) steps. Because of this possibility, these problems are usually called *non-deterministic polynomial* (NP, for short), and the class of such problems is denoted by NP.

Most computer scientists believe that there are problems in the class NP which cannot be solved in polynomial time, i.e., that NP$\neq$P; however, this has not been proven yet.

Not all the problems from the class of NP are of the same complexity. Some problems from the class NP – e.g., the problem of solving systems of linear equations – are relatively easy in the sense that they can be solved by polynomial time algorithms. Some problem are more difficult than others – because we can reduce every particular case of the first problem to special cases of the second problem. For example, we can reduce solution of a system of linear equations to solving quadratic equations (with 0 coefficients at x_i^2); this means that the general problem of solving systems of quadratic equations

is more difficult (or at least not less difficult) than the general problem of solving systems of linear equations.

Some general problems from the class NP are known to be the most difficult ones, in the sense that every other problem from the class NP can be reduced (in the above sense) to this particular problem. Such problems are called *NP-complete*. A similar notion of complexity can be extended to problems outside the class NP, for which we may not know how to check the correctness of the proposed solution in polynomial time. If any problem from the class NP can be reduced to such a problem, then this problem is called *NP-hard*. In these terms, a problem is NP-complete if it is NP-hard and belongs to the class NP.

It is worth mentioning that even if a general problem is NP-hard, its particular instance may be easy to solve. There may be efficient algorithms which solve particular instances from an important subclass of an NP-hard problem, there may be efficient heuristics which, in many cases, solve these problems.

Let us now formulate our results in precise terms. Some of these results were previously presented in [11].

5 First Theoretical Result: Detecting Decomposable Aggregability Is NP-Hard

Let us start by presenting the (known) formal definitions of a linear dynamical system.

Definition 1. *Let n be an integer. This integer will be called the* number of microvariables *(or* variables*, for short). These variables will be denoted by $x_1, \ldots, x_n$.*

Definition 2. *Let the number of microvariables n be given. By a* microstate *(or* state*), we mean an n-dimensional vector $x = (x_1, \ldots, x_n)$. By a* trajectory*, we means a function which maps integers t into states $x(t)$. The state $x(t)$ is called* a state at moment t.

Definition 3. *For a given n, by a* linear dynamical system*, we mean an $n \times n$ rational-valued matrix c with entries $c_{i,j}$, $1 \le i \le n$, $1 \le j \le n$. We say that a trajectory $x(t)$ is* consistent *with the linear dynamical system $c_{i,j}$ if for every t, we have*

$$x_i(t+1) = \sum_{j=1}^{n} c_{i,j} \cdot x_j(t). \tag{9}$$

Comment. In reality, the coefficients $c_{i,j}$ can be real numbers (not necessarily rational). However, our main objective is to analyze the corresponding algorithms. So, instead of the actual (unknown) value of each coefficient, we can only consider the (approximate) value represented in the computer, and in the computer, usually, only rational numbers are represented.

Now that we have reminded the reader about the formal definition of a dynamical system, let us formally describe what it means for a system to be decomposably aggregable. The main objective of this section is to prove that detecting decomposable aggregability is NP-hard even in the simplest case when we only have two blocks. In view of this objective, to simplify our notations, let us present the definitions of decomposable aggregability only for this case (when we only have 2 blocks I_1 and I_2).

To simplify the situation even more, let us fix a microvariable x_{i_0} (i.e., an index from 1 to n). In a partition, this microvariable can belong either to the first block or to the second block. Without losing generality, let us assume that it belongs to the first block I_1 (if it belong to the second block, we can simply rename the blocks).

Since i_0 belongs to the first block and the blocks do not overlap, only the first macrovariable y_1 can depend on this microvariable. We would also like to avoid a degenerate case in which the macrovariables do not depend on this microvariable x_{i_0} at all, so we require that y_1 actually depend on x_{i_0}. Let us describe this requirement in precise terms.

Definition 4. *Let us fix an index $i_0 \le n$.*

- *By a* 2-partition*, we mean a pair (I_1, I_2) of non-empty sets $I_1 \subseteq \{1, \ldots, n\}$ and $I_2 \subseteq \{1, 2, \ldots, n\}$ such that $i_0 \in I_1$, $I_1 \cup I_2 = \{1, \ldots, n\}$, and $I_1 \cap I_2 = \emptyset$.*
- *By* decomposable aggregation*, we mean a triple (I_1, I_2, α), where (I_1, I_2) is a 2-partition, and $\alpha = (\alpha_1, \ldots, \alpha_n)$ is a tuple for for which $\alpha_{i_0} \ne 0$.*
- *For every microstate $x = (x_1, \ldots, x_n)$, by the corresponding* macrostate *we mean a pair $y = (y_1, y_2)$, where $y_a = \sum\limits_{i \in I_a} \alpha_i \cdot x_i$.*

Definition 5. *We say that a decomposable aggregation (I_1, I_2, α) is* consistent *with the linear dynamical system $c_{i,j}$ if for every moment of time t, when two microstates $x(t)$ and $\widetilde{x}(t)$ correspond to the same macrostate $y(t) = \widetilde{y}(t)$, then in the next moment of time, the resulting microstates $x_i(t+1) = \sum\limits_{j=1}^{n} c_{i,j} \cdot x_j(t)$ and $\widetilde{x}_i(t+1) = \sum\limits_{j=1}^{n} c_{i,j} \cdot \widetilde{x}_j(t)$ also lead to the same macrostate: $y(t+1) = \widetilde{y}(t+1)$.*

Comment. In other words, we require that the next macrostate $y(t+1)$ is uniquely determined by the previous macrostate $y(t)$ (and does not depend on a specific microstate $x(t) = (x_1(t), \ldots, x_n(t))$. Since we only consider linear systems, this definition is equivalent to the following:

Definition 6. *We say that a decomposable aggregation (I_1, I_2, α) is* consistent *with the linear dynamical system $c_{i,j}$ if there exist values $h_{a,b}$ ($a = 1, 2$, $b = 1, 2$) such that for every $x_1, \ldots, x_n$, $x'_i = \sum\limits_{j=1}^{n} c_{i,j} \cdot x_j$ implies that*

$$y'_a = \sum_{b=1}^{2} h_{a,b} \cdot y_b, \tag{10}$$

where $y_a \stackrel{\text{def}}{=} \sum_{i \in I_a} \alpha_i \cdot x_i$ *and* $y'_a \stackrel{\text{def}}{=} \sum_{i \in I_a} \alpha_i \cdot x'_i$.

Definition 7. *Let $c_{i,j}$ be a linear dynamical system and let i_0 be a selected index. We say that the pair (c, i_0) is* (decomposably) 2-aggregable *if there exists a decomposable aggregation which is consistent with this linear dynamical system.*

Theorem 1. *Detecting decomposable 2-aggregability is NP-hard.*

Comments.

- The computational complexity of the problem comes from the fact that we are looking for decomposable aggregability, when the blocks I_a should be non-overlapping. For general aggregability (not necessarily decomposable), we allow overlapping blocks. In this case, we can easily find the corresponding combinations y_a, e.g., as the coordinates $y_a = \sum x_i \cdot e_i^{(a)}$ of the vector $x = (x_1, \ldots, x_n)$ in the basis formed by the eigenvectors $e^{(a)}$ of the matrix $c_{i,j}$.
- For the readers' convenience, the proofs of all our results are presented in the special (last) section.
- For readers who are more accustomed to matrix notations, a linear combination $\sum \alpha_i \cdot x_i$ can be represented as a matrix product $\alpha^T x$, where α^T denotes a transposition of the vector α.
- In the following text, we prove that for linear systems, once we have found the partition $I_1, \ldots, I_k$, we can then find the corresponding weights α_i in polynomial time. Thus, the problem of detecting decomposable aggregability belongs to the class NP; hence, our theorem actually implies that detecting decomposable 2-aggregability is NP-complete.

6 Approximate Aggregability: Possible Definitions

We have mentioned that in mathematical terms, the (exact) aggregability means that the dynamics of the macrovariables y_a is uniquely determined by the macrovariables themselves. For linear systems, this means that for each a, the value y'_a (that describes the dynamics of y_a) is equal to the expression $\sum_{b=1}^{k} h_{a,b} \cdot y_b$ determined only by the values of the macrovariables themselves.

In many practical cases, we do not have *exact* aggregability, we only have an *approximate* one. In other words, the difference Δy_a between the aggregation y'_a of x'_i and $\sum_{b=1}^{k} h_{a,b} \cdot y_b$ should be small.

The macrovariables are defined modulo multiplying by arbitrary scaling constants; so, in principle, for every real number $\delta > 0$, we consider the new macrovariables $Y_a \stackrel{\text{def}}{=} \delta \cdot y_a$. For these new macrovariables, the difference ΔY_a between Y'_a and $\sum_{b=1}^{k} h_{a,b} \cdot Y_b$ is equal to $\delta \cdot \Delta y_a$. Thus, no matter how large the original difference Δy_a is, the new difference ΔY_a can be arbitrarily small if we select an appropriate small value $\delta > 0$. Therefore, to come up with a meaningful definition of approximate aggregability, we must either provide a scale-independent criterion for approximate aggregability or, alternatively, assume that the macrovariables are appropriately scaled.

For linear systems, it is easy to scale the macrovariables in this manner. For example, once we fixed an index i_0, we can take the group that contains i_0 as I_1 (as we did before). Then, we can require that $\alpha_{i_0} = 1$ and that for every $a > 1$, we have $\alpha_i = 1$ for one of the values $i \in I_a$. Clearly, an arbitrary set of macrovariables Y_a can be thus rescaled if we replace Y_1 with $y_1 = Y_1/\alpha_{i_0}$ and Y_a $(a > 1)$ with $y_a = Y_a/\alpha_i$ for some $i \in I_a$.

In the following text, we assume that we are looking for thus rescaled macrovariables.

Since both y'_a and $\sum_{b=1}^{2} h_{a,b} \cdot y_b$ depend on the initial state x_i, the difference Δy_a is a function of the variables $x_1, \ldots, x_n$. In particular, for linear systems, Δy_a is a linear function of x_i, i.e., $\Delta y_a = \sum_{i=1}^{n} c_{ai} \cdot x_i$ for some coefficients c_{ai}.

A natural way to describe the smallness of the difference Δy_a is explained in [10] and [20]. Namely, we assume that we know the probability of different possible initial states. It is reasonable to assume that this probability distribution is non-degenerate, i.e., that it is not concentrated on a single surface within the n-dimensional space with probability 1.

In this case, it is natural to say that Δy_a is small if the mean value of its norm is small: $\int \|\Delta y\|^2 \cdot \rho(x)\, dx \le \varepsilon^2$, where $\varepsilon > 0$ is a given small number,

$$\|\Delta y\| = \|(\Delta y_1, \ldots, \Delta y_k)\| = \sqrt{\Delta y_1^2 + \ldots + \Delta y_k^2} \tag{11}$$

is the standard Euclidean (l^2-) norm of a vector $\Delta y = (\Delta y_1, \ldots, \Delta y_k)$, and $\rho(x)$ is the probability density corresponding to the actual distribution of the initial states.

In terms of the vector $c = \{c_{ai}\}$ of the coefficients c_{ai}, this definition of smallness can be described as $\|c\| \le \varepsilon$, where

$$\|c\| \stackrel{\text{def}}{=} \sqrt{\int \sum_{a=1}^{k} \left(\sum_{i=1}^{n} c_{ai} \cdot x_i \right)^2 \cdot \rho(x)\, dx}. \tag{12}$$

In some practical situations, we do not know the probabilities of different initial states. In such situations, we can define Δy_a to be small if, e.g., $|\Delta y_a| \le$

$\varepsilon \cdot \max_i |x_i|$ for all a. In terms of the vector c, this is equivalent to requiring that $\|c\| \le \varepsilon$, where

$$\|c\| \stackrel{\text{def}}{=} \max_{a,x} \frac{\left|\sum_{i=1}^{n} c_{ai} \cdot x_i\right|}{\max_i |x_i|}. \tag{13}$$

These two examples can be naturally generalized. Indeed, in both cases (12) and (13), we can check that the corresponding "smallness measure" $\|c\|$ is a *norm* on a linear space $\mathbb{R}^{k \cdot n}$ of all $(k \cdot n)$-dimensional vectors $c = \{c_{ai}\}$, in the sense that it satisfies the usual properties of the norm on a vector space (see, e.g., [4]):

- $\|c\| = 0$ if and only if $c_{ai} = 0$ for all a and i;
- for every real number λ and for every vector c, we have $\|\lambda \cdot c\| = |\lambda| \cdot \|c\|$;
- for every two vectors c and c', we have $\|c + c'\| \le \|c\| + \|c'\|$.

Thus, we can give a more general definition of smallness by requiring that $\|c\| \le \varepsilon$ for some norm $\|c\|$.

In some reasonable sense, the resulting definition of approximate aggregability does not depend on the choice of the norm. Indeed, it is known (see, e.g., [4]) that all the norms on a finite-dimensional linear space are equivalent, i.e., that for every two norms $\|\cdot\|_1$ and $\|\cdot\|_2$, there exist real numbers r_{12} and R_{12} such that for every vector c, we have

$$r_{12} \cdot \|c\|_1 \le \|c\|_2 \le R_{12} \cdot \|c\|_1. \tag{14}$$

Thus, if we use a norm $\|\cdot\|_1$ to define smallness, i.e., if we require that this norm does not exceed a threshold value ε ($\|c\|_1 \le \varepsilon$), then we can conclude that for this same vector c, the value $\|c\|_2$ of the second norm is also small: it does not exceed the value $\varepsilon' \stackrel{\text{def}}{=} R_{12} \cdot \varepsilon$. When the original threshold ε tends to 0, the new threshold $\varepsilon' = R_{12} \cdot \varepsilon$ tends to 0 as well.

Conversely, if we use a norm $\|\cdot\|_2$ to define smallness, i.e., if we require that this norm does not exceed a threshold value ε ($\|c\|_2 \le \varepsilon$), then we can conclude that for the same vector c, the value $\|c\|_1$ of the first norm is also small: it does not exceed the value $\varepsilon'' \stackrel{\text{def}}{=} R_{21} \cdot \varepsilon$. When the original threshold ε tends to 0, the new threshold $\varepsilon'' = R_{21} \cdot \varepsilon$ tends to 0 as well.

In view of this equivalence, in the following text, we will provide the detailed proof for only one norm. Proofs for other norms are similar.

Comment. Approximate aggregability means, crudely speaking, that for the same initial state $x_i(0)$, the values of the macrovariables $y_a(1)$ after a one-step iteration of the actual system are close to the result $y_a^{\text{aggr}}(1)$ of the one-step iteration of the corresponding aggregable system. From this, we can conclude that the values of the macrovariables $y_a(2)$ after a two-step iteration of the actual system are close to $y_a^{\text{aggr}}(2)$, and that the same is true for an arbitrary (fixed) number of steps t.

It should be mentioned, however, that in many important cases, the difference between the values $y_a(t)$ and $y^{\text{aggr}}(t)$ grows with time t. So, if we fix the initial state $x_i(0)$ and a threshold ε, then the values $y_a(t)$ and $y^{\text{aggr}}(t)$ are ε-close only until some critical time t_{critical}; after this critical time, the difference exceeds the desired threshold.

7 Second Theoretical Result: Detecting Approximate Decomposable Aggregability Is Also NP-Hard

Let us show that detecting *approximate* decomposable aggregability is also NP-hard, even in the simplest case of 2-aggregability ($k = 2$). As we have mentioned, in the linear case, macrovariables are defined modulo multiplying by arbitrary scaling constants; so, by applying appropriate "normalizing" re-scaling, we can safely assume, e.g., that $\alpha_{i_0} = 1$ for $i_0 \in I_1$ and that $\alpha_i = 1$ for some $i \in I_2$. Instead of requiring that the dynamics of the macrovariables y_a is uniquely determined by the macrovariables themselves, we can now require that the difference between y'_a and $\sum_{b=1}^{2} h_{a,b} \cdot y_b$ should be small, i.e., that the absolute value of this difference does not exceed $\varepsilon \cdot \max |x_i|$ for some small ε. We thus arrive at the following definition.

Definition 8. *We say that a decomposable aggregation (I_1, I_2, α) is* normalized *if $\alpha_{i_0} = 1$ and $\alpha_i = 1$ for some $i \in I_2$.*

Definition 9. *Let $\varepsilon > 0$ be a real number. We say that a normalized decomposable aggregation (I_1, I_2, α) is* ε-consistent *with the linear dynamical system $c_{i,j}$ if there exist values $h_{a,b}$ ($a = 1, 2$, $b = 1, 2$) such that for every $x_1, \ldots, x_n$, $x'_i = \sum_{j=1}^{n} c_{i,j} \cdot x_j$ implies that $\left| y'_a - \sum_{b=1}^{2} h_{a,b} \cdot y_b \right| \le \varepsilon \cdot \max_i |x_i|$, where*

$$y_a \stackrel{\text{def}}{=} \sum_{i \in I_a} \alpha_i \cdot x_i, \quad y'_a \stackrel{\text{def}}{=} \sum_{i \in I_a} \alpha_i \cdot x'_i. \tag{15}$$

Definition 10. *Let $c_{i,j}$ be a linear dynamical system, let i_0 be a selected index, and let $\varepsilon > 0$ be a real number. We say that the pair (c, i_0) is* ε-approximately (decomposably) 2-aggregable *if there exists a decomposable aggregation which is ε-consistent with this linear dynamical system.*

Theorem 2. *Detecting approximate decomposable 2-aggregability is NP-hard.*

Comment. We have already mentioned that all norms on $\mathbb{R}^n$ are equivalent. Thus, the proof of our result can be easily modified into a proof that we if we use other norm-based definitions of approximate decomposable aggregability, e.g., for the definition from [10], then detecting approximate decomposable aggregability is also NP-hard.

8 Once We Find the Partition, Finding the Combinations is Feasible

8.1 Algorithm

We have shown that finding the partition is NP-hard. Our original problem was not only to find a partition, but also to find the appropriate combinations y_a. Let us show that, in the linear case, once the partition is found, finding the weights of the corresponding combinations y_a is easy.

Indeed, in matrix terms, the original dynamic equation has the form $x' = cx$. Once the partition $I_1, \ldots, I_k$ is fixed, we can represent each n-dimensional state vector x as a combination of vectors $x^{(a)}$ formed by the components x_i, $i \in I_a$. In these terms, the equation $x' = cx$ can be represented as $x'^{(a)} = \sum\limits_b c^{(a),(b)} x^{(b)}$, where $c^{(a),(b)}$ denotes the corresponding block of the matrix c (formed by elements $c_{i,j}$ with $i \in I_a$ and $j \in I_b$). For the corresponding linear combinations $y_a = \alpha^{(a)\,T} x^{(a)}$, the dynamics takes the form $y'_a = \sum\limits_b \alpha^{(a)\,T} c^{(a),(b)} x^{(b)}$. The only possibility for this expression to only depend on the combinations $y_b = \alpha^{(b)\,T} x^{(b)}$ is when for each b, the coefficients of the dependence of y'_a on x_i, $i \in I_b$, are proportional to the corresponding weights α_i, i.e., when for every a and b, we have $\alpha^{(a)\,T} c^{(a),(b)} = \lambda_{a,b} \alpha^{(b)\,T}$ for some number $\lambda_{a,b}$. By transposing this relation, we conclude that

$$c^{(a),(b)\,T} \alpha^{(a)} = \lambda_{a,b} \alpha^{(b)}. \tag{16}$$

In particular, for $a = b$, we conclude that $\alpha^{(a)}$ is an eigenvector of the matrix $c^{(a),(a)\,T}$. Since the weight vectors $\alpha^{(a)}$ are defined modulo a scalar factor, we can thus select one of the (easy-to-compute) eigenvectors of $c^{(a),(a)\,T}$ as $\alpha^{(a)}$.

Once we know $\alpha^{(a)}$ for one a, we can determine all other weight vectors $\alpha^{(b)}$ from the condition (16), i.e., as $\alpha^{(b)} = c^{(a),(b)\,T} \alpha^{(a)}$.

Comment. A similar – but more complex – algorithms can be used in the nonlinear case as well. We will describe this nonlinear generalization in detail in a forthcoming paper.

8.2 Calculating aggregate dynamics

Once we know the partitions I_a and the corresponding weights $y_a = \sum\limits_{i \in I_a} \alpha_i \cdot x_i$, we can use the results from [20] to calculate the coefficients $h_{a,b}$ that describe the aggregate dynamics $y'_a = \sum\limits_b h_{a,b} \cdot y_b$. Namely, if we describe a $k \times n$ matrix α as $\alpha_{a,i} = \alpha_i^{(a)}$ if $i \in I_a$ and 0 otherwise, then $h = \alpha c \alpha^T (\alpha \alpha^T)^{-1}$.

8.3 Example

We now illustrate this algorithm on the above simple example, i.e., selection-mutation system where the fitness of each genotype depends only on the number of 0s and 1s.

It is assumed that we already know the partition of microvariables into clusters $I_1 = \{0\ldots0\}$, $I_2 = \{10\ldots0, 010\ldots0, \ldots, 0\ldots01\}$, etc. For each of these clusters, we consider the linear space whose dimension is equal to the number of microvariables in each cluster: the linear space corresponding to I_1 has dimension 1, the linear space corresponding to cluster I_2 has dimension g, the linear space corresponding to cluster I_3 has dimension $\frac{g\cdot(g-1)}{2}$, etc. The simplest, thus, easiest-to-analyze space is the space corresponding to I_1. For this space, the corresponding 1×1 matrix $c_{ij}^{(1),(1)}$ consists of the single element $1-g\cdot\mu$, and so, its eigenvector is the 1-element vector $\alpha_i = \alpha_{0\ldots0} = 1$.

According to our algorithm, we can now find the weights α_i for $i \in I_2$ by considering equation (16) for the matrix $c_{ij}^{(2),(1)}$. For each genotype $i \in I_2$ with one "1" and for the genotype $i \in I_1$ with all 0s, the term $c_{ij}\cdot x_j$ in the expression for $x_i' = x_i(t+1)$ is the same, i.e., $c_{ij}^{(2),(1)} = \text{const}$. Thus, because of the formula (16), all the elements i for $i \in I_2$ have the same weight: $\alpha_i = \text{const}$. So, the combination $y_2 = \sum\limits_{i\in I_2} \alpha_i\cdot x_i$ is proportional to the sum $\sum\limits_{i\in I_2} x_i$.

Similarly, by applying the formula (16) to the matrix $c_{ij}^{(3),(2)}$, we conclude that the weights α_i of all the genotypes $i \in I_3$ are also equal to each other, i.e., that the combination $y_3 = \sum\limits_{i\in I_3} \alpha_i\cdot x_i$ is also proportional to the sum $\sum\limits_{i\in I_3} x_i$, etc. Eventually, we will conclude that for each a, the macrovariable y_a is proportional to $\sum\limits_{i\in I_a} x_i$ – i.e., we get exactly the aggregation which is, as we have shown, consistent with the system's dynamics.

In this example, In this example, the matrix formula for h in terms of α and c [20] discussed in the previous subsection leads to the desired aggregate dynamics from Section 2.

8.4 What to Do in the Case of Approximate Aggregation

In the previous sections, we assumed that we know the partition for which an exact aggregation is possible, and we described n algorithm for finding the weights (i.e., macrovariables) corresponding to the exact aggregation. In practice, the aggregation is rarely exact. So, a more realistic setting is as follows: we know the partition, we must find the macrovariables that provide the approximate aggregation.

A natural way to proceed is to assume that the aggregation is exact and use the above algorithm to find the weights corresponding to this assumption.

It is reasonable to apply this heuristic algorithm for all the approximate aggregation situations. Moreover, it has been shown that for many reasonable continuous-time [9, 10] and discrete-time systems ([20], pp. 40–41), this approach leads to the best possible approximation.

9 Proofs

9.1 Proof of Theorem 1

As we have mentioned earlier, NP-hardness of a general problem means that any problem from the class NP can be reduced to this problem. Some problems are already known to be NP-hard. An example of such a problem is a *subset problem* (see, e.g., [7]). The subset problem is as follows: given n positive integers $s_1, \ldots, s_m$ and a positive integer s_0, whether there exists a subset $I \subseteq \{1, \ldots, m\}$ such that $\sum\limits_{i \in I} s_i = s_0$.

In order to prove that our problem is NP-hard, it is therefore sufficient to show that the subset problem can be reduced to our problem. Since we already know that every problem from the class NP can be reduced to the subset problem, and the subset problem can be reduced to our problem, then we will be able to conclude that an arbitrary problem from the class NP can be reduced to our problem – i.e., that our problem is indeed NP-hard.

Therefore, to prove our theorem, it is sufficient to prove that the subset problem can be reduced to the problem of checking 2-aggregability, i.e., that for every instance of the subset problem, we can find an equivalent instance of the 2-aggregation problem. Let us show how this can be done.

The subset problem does not have a solution if $s_0 > \sum\limits_{i=1}^{m} s_i$, so it is sufficient to consider only instances of the subset problem for which $s_0 \le \sum\limits_{i=1}^{m} s_i$. When $s_0 = \sum\limits_{i=1}^{m} s_i$, then $I = \{1, \ldots, m\}$ is an obvious solution. Thus, it is sufficient to only consider instances of this problem for which $s_0 < \sum\limits_{i=1}^{m} s_i$, i.e. (since s_0 and s_i are integers), for which $s_0 \le \sum\limits_{i=1}^{m} s_i - 1$.

For every instance of the subset problem that satisfies this condition, we take $s_{m+1} \stackrel{\text{def}}{=} -s_0$, $n = m + 2$, $i_0 = m + 2$, and we form the following linear system:

- for $1 \le i \le m+1$, we take $c_{i,i} = 1$, $c_{i,n+1} = s_i$, and $c_{i,j} = 0$ for all other j;
- for $i = m + 2$, we take $c_{m+2,m+2} = 1 + \beta$, where $\beta \stackrel{\text{def}}{=} 1 + s_0 - \sum\limits_{i=1}^{m} s_i$, and $c_{m+2,i} = 1$ for all other i.

Since $s_0 \le \sum_{i=1}^{m} s_i - 1$, we have $\beta \le 0$.

Let us prove that this system is consistent if and only if the original instance of the subset problem has a solution.

"If" part.

If the original instance has a solution I, with $CI \stackrel{\text{def}}{=} \{1, \ldots, m\} - I$, then we take $I_1 = CI \cup \{m+2\}$, $I_2 = I \cup \{m+1\}$, $\alpha_i = 1$ for all i, $h_{1,1} = 2$, $h_{1,2} = h_{2,2} = 1$, and $h_{2,1} = 0$. Then, we should have $y_1' = 2y_1 + y_2$ and $y_2' = y_2$.

Indeed, the fact that I is a solution means that $\sum_{i \in I} s_i = s_0$. For our choice of weights α_i, we get $y_1 = \sum_{i \in CI} x_i + x_{m+2}$ and $y_2 = \sum_{i \in I} x_i + x_{m+1}$. For y_2', we get

$$y_2' = \sum_{i \in I} x_i' + x_{m+1}' = \sum_{i \in CI} x_i + x_{m+1} + \left(\sum_{i \in I} -s_0 \right) \cdot x_{m+2}. \tag{17}$$

Since $\sum_{i \in I} s_i = s_0$, we conclude that $y_2' = y_2$.

Similarly,

$$y_1' = \sum_{i \in CI} x_i' + x_{m+2}'. \tag{18}$$

Describing the sum $\sum_{i=1}^{m+2} x_i$ in the expression for x_{m+2}' as the sum of the values from I, CI, $m+1$, and $m+2$, we conclude that

$$y_1' = \sum_{i \in CI} x_i + \left(\sum_{i \in CI} s_i \right) \cdot x_{m+2} + \sum_{i \in CI} x_i + \sum_{i \in I} x_i + x_{m+1} + x_{m+2} + \beta \cdot x_{m+2} =$$

$$\left(\sum_{i \in I} x_i + x_{m+1} \right) + 2 \cdot \sum_{i \in CI} x_i + \left(\sum_{i \in CI} s_i + 1 + \beta \right) \cdot x_{m+2}. \tag{19}$$

Since $\sum_{i \in I} s_i = s_0$, we have $\sum_{i \in CI} s_i = \sum_{i=1}^{m} s_i - \sum_{i \in I} s_i = \sum_{i=1}^{m} s_i - s_0$. Due to our choice of β, we thus have $\sum_{i \in CI} s_i + 1 + \beta = 2$, hence $y_1' = 2y_1 + y_2$.

"Only if" part.

Conversely, let us assume that the system is 2-aggregable, i.e., that there exist sets $I_1 \ni m+2$, and the values α_i and $h_{a,b}$ for which all the above conditions are satisfied. In other words, $\alpha_{m+2} \ne 0$, and from the equations

$$x_i' = x_i + s_i \cdot x_{m+2}, \quad 1 \le i \le m+1, \tag{20}$$

$$x'_{m+2} = \sum_{i=1}^{m+2} x_i + \beta \cdot x_{m+2}, \tag{21}$$

we should be able to conclude that for

$$y_1 = \sum_{i \in I_1} \alpha_i \cdot x_i, \quad y_2 = \sum_{i \in I_2} \alpha_i \cdot x_i, \tag{22}$$

$$y'_1 = \sum_{i \in I_1} \alpha_i \cdot x'_i, \quad y'_2 = \sum_{i \in I_2} \alpha_i \cdot x'_i, \tag{23}$$

we have

$$y'_1 = h_{1,1} \cdot y_1 + h_{1,2} \cdot y_2 \tag{24}$$

and

$$y'_2 = h_{2,1} \cdot y_1 + h_{2,2} \cdot y_2. \tag{25}$$

Let us denote $I'_1 \stackrel{\text{def}}{=} I_1 - \{m+2\}$. From (20), (21), and (23), we conclude that

$$y'_1 = \sum_{i \in I'_1} \alpha_i \cdot x_i + \left(\sum_{i \in I'_1} \alpha_i \cdot s_i \right) \cdot x_{m+2} + \alpha_{m+2} \cdot \sum_{i=1}^{m+2} x_i + \alpha_{m+2} \cdot \beta \cdot x_{m+2}. \tag{26}$$

Thus, the equation (24) takes the form

$$\sum_{i \in I'_1} \alpha_i \cdot x_i + \left(\sum_{i \in I'_1} \alpha_i \cdot s_i \right) \cdot x_{m+2} + \alpha_{m+2} \cdot \sum_{i=1}^{m+2} x_i + \alpha_{m+2} \cdot \beta \cdot x_{m+2} =$$

$$h_{1,1} \cdot \left(\sum_{i \in I'_1} \alpha_i \cdot x_i + \alpha_{m+2} \cdot x_{m+2} \right) + h_{1,2} \cdot \sum_{i \in I_2} \alpha_i \cdot x_i. \tag{27}$$

Since this equality must hold for all possible values of x_i, for each i, the coefficient at x_i in the left-hand side of (27) must be equal to the coefficient at x_i in the right-hand side of (27).

In particular, for $i \in I'_1$, we conclude that $\alpha_i + \alpha_{m+2} = h_{1,1} \cdot \alpha_i$, i.e., that $(h_{1,1} - 1) \cdot \alpha_i = \alpha_{m+2}$. Since $\alpha_{m+2} \neq 0$, we conclude that $\alpha_i \neq 0$ and $h_{1,1} - 1 \neq 0$, hence $\alpha_i = \alpha_{m+2}/(h_{1,1} - 1)$ for all such i – i.e., all the values α_i, $i \in I'_1$ are equal to each other. Let us denote the common value of these α_i by $\alpha^{(1)} \neq 0$.

For $i \in I_2$, we similarly conclude that $\alpha_{m+2} = h_{1,2} \cdot \alpha_i$. Since $\alpha_{m+2} \neq 0$, we similarly conclude that $\alpha_i \neq 0$, and that $\alpha_i = \alpha_{m+2}/h_{1,2}$ is the same for all $i \in I_2$. Let us denote the common value of these α_i by $\alpha^{(2)} \neq 0$. Thus, the formulas (22)–(23) take the following form:

$$y_1 = \alpha^{(1)} \cdot \sum_{i \in I'_1} x_i + \alpha_{m+2} \cdot x_{m+2}, \quad y_2 = \alpha^{(2)} \cdot \sum_{i \in I_2} x_i, \tag{28}$$

$$y'_1 = \alpha^{(1)} \cdot \sum_{i \in I'_1} x'_i + \alpha_{m+2} \cdot x'_{m+2}, \quad y'_2 = \alpha^{(2)} \cdot \sum_{i \in I_2} x'_i. \tag{29}$$

By using the expressions (20)–(22) and (9a), we conclude that y'_2 takes the form

$$y'_2 = \alpha^{(2)} \cdot \sum_{i \in I_2} x_i + \alpha^{(2)} \cdot \left(\sum_{i \in I_2} s_i \right) \cdot x_{m+2}. \tag{30}$$

Thus, the equation (25) takes the form

$$\alpha^{(2)} \cdot \sum_{i \in I_2} x_i + \alpha^{(2)} \cdot \left(\sum_{i \in I_2} s_i \right) \cdot x_{m+2} =$$

$$h_{2,1} \cdot \left(\alpha^{(1)} \cdot \sum_{i \in I'_1} x_i + \alpha_{m+2} \cdot x_{m+2} \right) + h_{2,2} \cdot \alpha^{(2)} \cdot \sum_{i \in I_2} x_i. \tag{31}$$

For $i \in I'_1$, by equating coefficients at x_i in both sides of (31), we conclude that $h_{2,1} \cdot \alpha^{(1)} = 0$. Since $\alpha^{(1)} \neq 0$, we thus conclude that $h_{2,1} = 0$. By comparing coefficients at x_{m+2}, we now conclude that $\alpha^{(2)} \cdot \sum\limits_{i \in I_2} s_i = 0$. Since $\alpha^{(2)} \neq 0$, we thus conclude that $\sum\limits_{i \in I_2} s_i = 0$. Since all the values s_i are positive except for $s_{m+1} = -s_0$, the only possibility to have $\sum\limits_{i \in I_2} = 0$ is when $m+1 \in I_2$. In this case, for $I \stackrel{\text{def}}{=} I_2 - \{m+1\}$, we get $\sum\limits_{i \in I} s_i + (-s_0) = 0$, i.e., $\sum\limits_{i \in I} s_i = s_0$. So, the original instance of the subset problem has a solution.

This completes the proof of the theorem.

9.2 Proof of Theorem 2

To prove NP-hardness of approximate 2-aggregability, we use the same reduction to subset problem that we used in Theorem 1. Namely, for every instance of the subset problem, we define the same linear system: as in the previous proof; we will show that this system is approximately 2-aggregable if and only if the original instance of the subset problem has a solution.

We have already shown that if the original instance has a solution I, then the corresponding system is 2-aggregable and therefore, approximately 2-aggregable.

Conversely, let us assume that the system is approximately 2-aggregable, i.e., that there exist sets $I_1 \ni m+2$ and I_2, and the values α_i and $h_{a,b}$ for which all the above conditions are satisfied. In other words, $\alpha_{m+2} = 1$, and from the equations (20) and (21), we should be able to conclude that for the values y_a and y'_a we have

$$|y'_1 - h_{1,1} \cdot y_1 - h_{1,2} \cdot y_2| \leq \varepsilon \cdot \max_i |x_i| \tag{32}$$

and

$$|y_2' - h_{2,1} \cdot y_1 - h_{2,2} \cdot y_2| \le \varepsilon \cdot \max_i |x_i|. \tag{33}$$

By using the expression for y_1' from the previous proof, and taking into account that $\alpha_{m+2} = 1$, we conclude that the condition (32) takes the form

$$\left| \sum_{i \in I_1'} \alpha_i \cdot x_i + \left(\sum_{i \in I_1'} \alpha_i \cdot s_i \right) \cdot x_{m+2} + \sum_{i=1}^{m+2} x_i + \beta \cdot x_{m+2} - \right.$$

$$\left. h_{1,1} \cdot \left(\sum_{i \in I_1'} \alpha_i \cdot x_i + x_{m+2} \right) - h_{1,2} \cdot \sum_{i \in I_2} \alpha_i \cdot x_i \right| \le \varepsilon \cdot \max_i |x_i|. \tag{34}$$

This inequality must hold for all possible values of x_i. In particular, for each i, we can take $x_i = 1$ and $x_j = 0$ for all $j \ne i$, and conclude that the coefficient at x_i at the left-hand side of (34) should not exceed ε.

In particular, for $i \in I_1'$, we conclude that

$$|\alpha_i \cdot (1 - h_{1,1}) + 1| \le \varepsilon. \tag{35}$$

For $i \in I_2$, we conclude that

$$|1 - h_{1,2} \cdot \alpha_i| \le \varepsilon, \tag{36}$$

and for $i = m + 2$, we conclude that

$$\left| \sum_{i \in I_1'} \alpha_i \cdot s_i + 1 + \beta - h_{1,1} \right| \le \varepsilon. \tag{37}$$

Let us first use (35) to prove, by reduction to a contradiction, that $|h_{1,1}| \le \sum_{i=1}^{m} s_i + 1$. In this proof, we will use the known fact that for every two numbers b and c, we have $||b| - |c|| \le |b + c| \le |b| + |c|$.

Indeed, suppose that $|h_{1,1}| > \sum_{i=1}^{m} s_i + 1$; then $|1 - h_{1,1}| \ge ||h_{1,1}| - 1| = \sum_{i=1}^{m} s_i$. From (35), we conclude that $|\alpha_i| \cdot |1 - h_{1,1}| \le 1 + \varepsilon$, hence $|\alpha_i| \le \dfrac{1 + \varepsilon}{\sum_{i=1}^{m} s_i}$. Thus,

$$\left| \sum_{i=1}^{m} \alpha_i \cdot s_i \right| \le \sum_{i=1}^{m} |\alpha_i| \cdot s_i \le \sum_{i=1}^{m} s_i \cdot \frac{1 + \varepsilon}{\sum_{i=1}^{m} s_i} = 1 + \varepsilon. \tag{38}$$

Since we assumed that $|h_{1,1}| > \sum_{i=1}^{m} s_i + 1$ and we know that $\beta \le 0$, we conclude that

$$\left|h_{1,1} - 1 - \beta - \sum_{i=1}^{m} \alpha_i \cdot s_i\right| > \left(\sum_{i=1}^{m} s_i + 1\right) - 1 - (-\beta) - (1+\varepsilon). \qquad (39)$$

Substituting $\beta = 1 + s_0 - \sum_{i=1}^{m}$ into this expression, we conclude that

$$\left|h_{1,1} - \beta - \sum_{i=1}^{m} \alpha_i \cdot s_i\right| > s_0 - \varepsilon. \qquad (40)$$

Since s_0 is a positive integer, we have $s_0 - \varepsilon > 1 - \varepsilon$; so for $\varepsilon < \frac{1}{2}$, we get $s_0 - \varepsilon > \frac{1}{2} > \varepsilon$ and thus, $\left|h_{1,1} - 1 - \beta - \sum_{i=1}^{m} \alpha_i \cdot s_i\right| > \varepsilon$, which contradicts to the inequality (37).

This contradiction shows that $|h_{1,1}|$ cannot be larger than $\sum_{i=1}^{m} s_i + 1$, so $|h_{1,1}| \le \sum_{i=1}^{m} s_i + 1$. In this case,

$$|1 - h_{1,1}| \le \sum_{i=1}^{n} s_i + 2. \qquad (41)$$

From the equation (35), we conclude that $|\alpha_i \cdot (1 - h_{1,1})| \ge 1 - \varepsilon$, and therefore, due to (41), that

$$|\alpha_i| \ge \frac{1-\varepsilon}{\sum_{i=1}^{n} s_i + 2}. \qquad (42)$$

Let us now analyze the consequences of the condition (36). We have requested that for some $i \in I_2$, we have $\alpha_i = 1$. For this i, the condition (36) takes the form $|1 - h_{1,2}| \le \varepsilon$, i.e.,

$$1 - \varepsilon \le h_{1,2} \le 1 + \varepsilon. \qquad (43)$$

For all other i, we get

$$1 - \varepsilon \le h_{1,2} \cdot \alpha_i \le 1 + \varepsilon. \qquad (44)$$

Dividing (44) by (43), we conclude that for all $i \in I_2$, we have

$$\frac{1-\varepsilon}{1+\varepsilon} \le \alpha_i \le \frac{1+\varepsilon}{1-\varepsilon}, \qquad (45)$$

hence

$$-\frac{2\varepsilon}{1+\varepsilon} \le \alpha_i - 1 \le \frac{2\varepsilon}{1-\varepsilon}, \qquad (46)$$

and

$$|\alpha_i - 1| \le \frac{2\varepsilon}{1-\varepsilon}. \tag{47}$$

For y_2', we have

$$y_2' = \sum_{i \in I_2} \alpha_i \cdot x_i + \left(\sum_{i \in I_2} \alpha_i \cdot s_i\right) \cdot x_{m+2}, \tag{48}$$

and therefore, taking into account that $\alpha_{m+2} = 1$, we can describe the condition (33) in the form

$$\left|\sum_{i \in I_2} \alpha_i \cdot x_i + \left(\sum_{i \in I_2} \alpha_i \cdot s_i\right) \cdot x_{m+2} - \right.$$

$$\left. h_{2,1} \cdot \left(\sum_{i \in I_1'} \alpha_i \cdot x_i + x_{m+2}\right) - h_{2,2} \cdot \sum_{i \in I_2} \alpha_i \cdot x_i \right| \le \varepsilon \cdot \max_i |x_i|. \tag{49}$$

Similarly to the previous case, we conclude that the coefficient at x_i at the left-hand side of (49) should not exceed ε. So, for $i \in I_1'$, we conclude that

$$|h_{2,1} \cdot \alpha_i| \le \varepsilon. \tag{50}$$

For $i \in I_2$, we conclude that

$$|\alpha_i - h_{2,2} \cdot \alpha_i| \le \varepsilon, \tag{51}$$

and for $i = m + 2$, we conclude that

$$\left|\sum_{i \in I_2} \alpha_i \cdot s_i - h_{2,1}\right| \le \varepsilon. \tag{52}$$

From (50) and (42), we conclude that

$$|h_{2,1}| = \frac{|h_{2,1} \cdot \alpha_i|}{|\alpha_i|} \le \frac{\varepsilon}{1-\varepsilon} \cdot \left(\sum_{i=1}^{m} s_i + 2\right). \tag{53}$$

The condition (52) can be rewritten as

$$\left|\sum_{i \in I_2} s_i + \sum_{i \in I_2} (\alpha_i - 1) \cdot s_i - h_{2,1}\right| \le \varepsilon, \tag{54}$$

hence

$$\left|\sum_{i \in I_2} s_i\right| \le \sum_{i \in I_2} |\alpha_i - 1| \cdot s_i + |h_{2,1}| + \varepsilon. \tag{55}$$

Using (47) and (53), we conclude that

$$\left|\sum_{i\in I_2} s_i\right| \le \frac{2\varepsilon}{1-\varepsilon}\cdot\sum_{i\in I_2} s_i + \frac{\varepsilon}{1-\varepsilon}\cdot\left(\sum_{i=1}^{m} s_i + 2\right) + \varepsilon. \tag{56}$$

Replacing the sum $\sum\limits_{i\in I_2} s_i$ in the right-hand side with a larger sum $\sum\limits_{i=1}^{m} s_i$, we get

$$\left|\sum_{i\in I_2} s_i\right| \le \frac{2\varepsilon}{1-\varepsilon}\cdot\sum_{i=1}^{m} s_i + \frac{\varepsilon}{1-\varepsilon}\cdot\left(\sum_{i=1}^{m} s_i + 2\right) + \varepsilon. \tag{57}$$

If $\varepsilon < \dfrac{1}{8\cdot\sum\limits_{i=1}^{m} s_i}$, then $\varepsilon < \dfrac{1}{8}$, hence $1-\varepsilon > \dfrac{7}{8}$ and $\dfrac{1}{1-\varepsilon} < \dfrac{8}{7}$. Hence, the condition (57) implies that

$$\left|\sum_{i\in I_2} s_i\right| \le 2\varepsilon\cdot\frac{8}{7}\cdot\sum_{i=1}^{m} s_i + \varepsilon\cdot\frac{8}{7}\cdot\sum_{i=1}^{m} s_i + \frac{8}{7}\cdot(2\varepsilon) + \varepsilon. \tag{58}$$

Because of our choice of ε, the first two terms in the right-hand side of (58) can be bounded as follows:

$$3\cdot\frac{8}{7}\cdot\left(\varepsilon\cdot\sum_{i=1}^{m} s_i\right) \le 3\cdot\frac{8}{7}\cdot\frac{1}{8} = \frac{3}{7} < \frac{1}{2}. \tag{59}$$

Similarly, since $\varepsilon < \dfrac{1}{8}$, we conclude that

$$\frac{8}{7}\cdot(2\varepsilon) + \varepsilon < \frac{8}{7}\cdot\left(2\cdot\frac{1}{8}\right) + \frac{1}{8} = \frac{2}{7} + \frac{1}{8} = \frac{23}{56} < \frac{1}{2}. \tag{60}$$

Therefore, (58) implies that

$$\left|\sum_{i\in I_2} s_i\right| < \frac{1}{2} + \frac{1}{2} = 1. \tag{61}$$

Since all the values s_i are integers, the sum $\sum\limits_{i\in I_2} s_i$ is also an integer, and the fact that the absolute value of this sum is smaller than 1 means that this sum must be equal to 0.

We have already shown, in the proof of Theorem 1, that this equality implies that the original instance of the subset problem has a solution. This completes the proof of the theorem.

10 Conclusion

In most biological system, we deal with a large number of microvariables. To simplify the analysis of such systems, it is desirable to reduce the dimensionality of the system's description by introducing *macrovariables* – combinations

of microvariables from different classes. From the practical viewpoint, it is therefore very important to identify possible aggregations of microvariables into macrovariables which are consistent with the system's dynamics.

In addition to the practical importance of reducing the complexity of the dynamical system, aggregation of the variables allows us to identify components of the system (subsets of variables) that are quasi-independent modules or otherwise have significant equivalence or symmetry properties. This aspects has been explored formally from the group-symmetry prospective in [17, 18].

Our main result shows, crudely speaking, that no general efficient algorithm is possible for identifying such aggregation. This means that, in general, for a system with n microvariables, we need $\approx 2^n$ computational steps (i.e., in effect, an exhaustive search) to find an appropriate aggregation. For biological systems, with a large number n of microvariables, such an exhaustive search is not possible. For example, for a genetic system with g possible loci with two allels, there are 2^g possible genotypes and thus, 2^{2^g} possible subsets of microvariables. Even for a relatively small number of loci $g = 10$, we need $2^{2^{10}} = 2^{1024} \approx 10^{300}$ computational steps – much more than the number of particles in the universe.

Thus, to efficiently aggregate a system, we must, in general, specify some additional information about the grouping of the variables. When we do have such information, then it is possible to efficiently generate the desired aggregations. In this chapter, we describe the corresponding algorithm for the case of linear systems. These results can be readily extended to nonlinear systems with differentiable state equations by local linearization at all states; we will explore this approach in a forthcoming paper.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, and by Texas Department of Transportation grant No. 0-5453.

The authors are thankful to Pierre-Jacques Courtois and Michael Vose for useful ideas, references, and suggestions, and to the anonymous referees for valuable suggestions.

References

1. Courtois PJ (1977) Decomposability: queueing and computer system applications. Academic Press, New York
2. Crow J, Kimura M (1970) An introduction to population genetics theory. Harper and Row, New York
3. Davidson EH (2006) The regulatory genome: gene regulatory networks in development and evolution. Academic Press, New York
4. Edwards RE (1995) Functional analysis: theory and applications. Dover, New York

5. Eigen M, McCaskill J, Schuster P (1989) The molecular quasispecies. Advances in Chemistry and Physics 75:149–263
6. Fell D (1996) Understanding the control of metabolism (Frontiers in metabolism). Ashgate Publ., London
7. Garey MR, Johnson DS (1979) Computers and intractability, a guide to the theory of np-completeness. WH Freeman and Company, San Francisco, California
8. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, Michigan
9. Iwasa Y, Andreasen V, Levin SA (1987) Aggregation in model ecosystems. I. Perfect aggregation. Ecological Modelling 37:287–302
10. Iwasa Y, Levin SA, Andreasen V (1989) Aggregation in model ecosystems. II. Approximate aggregation. IMA Journal of Mathematics Applied in Medicine and Biology 6:1–23
11. Kreinovich V, Shpak M (2006) Aggregability is NP-hard. ACM SIGACT, 37(3):97–104
12. Moey CCJ, Rowe JE (2004). Population aggregation based on fitness. Natural Computing 3(1):5–19.
13. Notredame C, Higgins DG (1996) SAGA: sequence alignment by genetic algorithm. Nucleic Acids Research 24:1515–24.
14. Notredame C, O'Brien EA, Higgins DG (1997) RAGA: RNA sequence alignment by genetic algorithm. Nucleic Acids Res. 25:4570–4580.
15. Reeves CR, Rowe JE (2002) Genetic algorithms: principles and perspectives. Kluwer Academic Publishers, Dordrecht
16. Rowe J (1998) Population fixed-points for functions of unitation. In: Reeves C, Banzhaf W (Eds), Foundations of Genetic Algorithms, Morgan Kaufmann Publishers, Vol. 5
17. Rowe JE, Vose MD, Wright AH (2005). Coarse graining selection and mutation. In: Proceedings of the 8th International Workshop on Foundations of Genetic Algorithms FOGA'2005, Aizu-Wakamatsu City, Japan, January 5–9, 2005, Springer Lecture Notes in Computer Science, Vol. 3469, pp. 176–191
18. Rowe JE, Vose MD, Wright AH (2005) State aggregation and population dynamics in linear systems. Artificial Life 11(4):473–492
19. Schuster P, Swetina J (1988) Stationary mutant distributions and evolutionary optimization. Bulletin of Mathematical Biology 50:635–650
20. Shpak M, Stadler PF, Wagner GP, Hermisson J (2004) Aggregation of variables and system decomposition: application to fitness landscape analysis. Theory in Biosciences 123:33–68
21. Shpak M, Stadler PF, Wagner GP, Altenberg L (2004). Simon-Ando decomposability and mutation-selection dynamics. Theory in Biosciences 123:139–180
22. Simon H, Ando F (1961) Aggregation of variables in dynamical systems. Econometrica 29:111–138
23. Stadler PF, Tinhofer G (2000). Equitable partitions, coherent algenras, and random walks: applications to the correlation structure of landscapes. MATCH 40:215–261