On the NP-Completeness of Computing the Commonality Among the Objects Upon Which a Collection of Agents Has Performed an Action

Roberto Alonso and Raúl Monroy

Department of Computer Science,
Tecnológico de Monterrey, Campus Estado de México,
Carr. lago de Guadalupe Km 3.5, Atizapán, Estado de México,
Mexico

{roberto.alonso, raulm}@itesm.mx

Abstract. We prove the NP-completeness of the so-called *Social Group Commonality* (SGC) problem which queries the commonality among the objects 'touched' by collections of agents while executing an action. Although it naturally arises in several contexts, *e.g.*, in profiling the behavior of a collection of system users, SGC (to the authors' knowledge) has been ignored. Our proof of SGC NP-completeness consists of a Karp reduction from the well-known Longest Common Subsequence (LCS) problem to SGC. We also prove that a special case of SGC which we call *2-SGC*, where the commonality among actions is limited to agent pairs, remains NP-complete. For proving NP-completeness of 2-SGC though, our reduction departs from the well-known Hitting Set problem. Finally, we hypothesize that the optimality version of SGC is NP-hard, hinting on how to deal with the proof obligation.

Keywords. Social Group Commonality, complexity theory, social networks, graphs.

Es NP-completo calcular la comunidad entre los objetos sobre los que una colección de agentes ha realizado una acción

Resumen. En este trabajo demostramos que el problema que llamamos *Comunalidad de grupos sociales* (SGC por sus siglas en inglés) es NP-completo. Este problema consulta la comunalidad entre los objetos tocados por una colección de agentes que ejecutan acciones. Aunque se presenta naturalmente en varios contextos *e.g.*, perfilar el comportamiento de un conjunto de usuarios de un sistema, SGC ha sido, acorde al conocimiento de los autores, ignorado. Nuestra demostración consiste en una reducción de Karp a partir del problema conocido como *Longest Common Subsequence* (LCS). Probamos también que un caso especial, al que llamamos *2-SGC*, donde la comunalidad entre las acciones está limitada a pares de agentes, sigue siendo NP-completo. Para probar 2-SGC, nuestra reducción parte del problema conocido como *Hitting Set*. Antes de concluir con el artículo, especulamos que la versión de optimización de SGC es *NP-duro*, dando indicaciones de como realizar la demostración necesaria.

Palabras clave. Comunalidad de grupos sociales, teoría de la complejidad, redes sociales, grafos.

1 Introduction

Grouping is an everyday task. We tend to group objects that are somehow similar, forming a class. Often, grouping is one-dimensional; for example, we put in the same class people with similar income, the same gender, or with comparable social status. Multi-dimensional grouping is much more interesting, although harder to compute. Indeed, there exists a number of daily situations, where we would like to know the commonality among the objects, upon which a collection of agents has executed some action. For instance, we would like to know all the connections (*e.g.*, friends, colleagues, etc.) that have in common a collection of users of a given social network (*e.g.*, Facebook, Linkedin, etc.) This problem which we call *Social Group Commonality* (SGC) is common, simple to formulate, and interestingly hard.

For example, SGC can be used to structure the use of a DNS server, so as to investigate how a collection of agents (an IP address together with a port number) relate to one another on the basis of the web domains (an URL) they have commonly visited over a period of time. We use this SGC formulation for the detection of a DNS attack, validating the working hypothesis that

ISSN 1405-5546
DOI: 10.13053/CyS-17-4-2013-003

malicious activity forms anomaly patterns in the groups linking users with URL's [1].

Another example of a SGC application consists of computing the semantic relatedness of text documents using wikification. Wikification is the process of relating words in an arbitrary text to concepts, structured as wikipedia articles [14, 11, 10]. We might like to know the relation (whether weak or strong) among these articles, following the idea that an article should be supported by a group of concepts. We could use this relation as a criterion for an article-concept ranking. Semantic meaning of wikipedia articles could be useful for numerous applications, including the semantic web. Works like [6] study the complexity of some aspects of the semantic web.

Also, forming customer classes is formulated as SGC. For example, the staff of sales or marketing would be eager to have all their customers grouped together in terms of the commonality of the goods or services they have somehow recently requested for. These could be used, for example, to elaborate sales offers, or issue a marketing campaign based on a finer customer profile. Other example applications include computing the file system objects that are used simultaneously by a collection of system users, the books or music commonly bought by a set of individuals and many others.

In this paper, we prove that SGC is actually NP-complete. Our proof consists of a Karp reduction from the well-known Longest Common Subsequence (LCS) problem to SGC. Roughly speaking, the LCS decision problem consists of determining whether there exists a string of at least a given length that is a subsequence of every string belonging to a finite set of strings. Not surprisingly, LCS has also a number of applications including bioinformatics and file comparison.

Interestingly, even computing the commonality among the objects referred to by the actions carried out by agent pairs is NP-complete, provided that the objects remain in a given object set. Our proof of the NP-completeness of this special SGC case which we call *2-SGC* also consists of a Karp reduction; this time, though, our reduction departs from the well-known Hitting Set (HS) problem to SGC. Roughly speaking, the HS decision problem consists of determining whether there exists a set of at least a given cardinality such that it contains at least one element which is also part of every set from a given collection of sets.

We conclude the paper hypothesizing that the optimality version of SGC is NP-hard.

The paper has the following structure. In Section 2 we formally introduce the *Social Group Commonality* problem, it also serves as a later reference to formulate the SGC and 2-SGC decision problems. Next, the Karp reduction (Section 3) is shown as a proof of the SGC NP-completeness. Section 4 presents the 2-SGC problem which is a special case of SGC; we also prove that it is NP-complete. Finally, remarks and indications for further work are given in Section 5.

2 Social Groups: Computing Commonality Among Users, Actions, and Objects

In this section, we provide a model for the behavior of a social group based on the objects referred to by an action carried out by a collection of agents. We shall use this model in order to formalize both decision problems: SGC and 2-SGC.

2.1 Agents Executing Actions over Objects for a Period of Time

Given that the behavior of an agent might change over time (*e.g.*, people may lose friends, books, etc.), the kinds of agent interactions, together with the kinds of agent relations these interactions give rise to, remain for only a period of time. Accordingly, we capture relations of interest relative to a given time period called a *window*. A window is given by a number of agent actions each of which we henceforth call a *query*.

Let $\mathbb{W}$ be the set of all windows, ranged over by $w_1, w_2, \ldots$, $\mathbb{A}$ the set of all agents, ranged over by $a_1, a_2, \ldots$, and let $\mathbb{O}$ be the set of all objects, ranged over by $o_1, o_2, \ldots$. We shall use $\mathrm{qry}^w(a, o)$ to denote that agent a has queried object o, over window w. The set of active agents, with respect to a given window $w \in \mathbb{W}$, is defined as follows:

$$\mathrm{agt}(w) = \{x \in \mathbb{A} \mid \exists y \in \mathbb{O}, \mathrm{qry}^w(x, y)\}$$

Likewise, the set of objects, onto which the agent actions have been performed, is given by

$$\mathrm{obj}(w) = \{y \in \mathbb{O} \mid \exists x \in \mathbb{A}, \mathrm{qry}^w(x, y)\}$$

Clearly, given a window $w \in$ W, the activity of agents, $\mathrm{agt}(w)$, over a collection of objects, $\mathrm{obj}(w)$, can be represented by means of a query matrix, Q^w, of size $|\mathrm{agt}(w)| \times |\mathrm{obj}(w)|$ (where $|S|$ denotes the cardinality of set S), such that $Q^w_{i,j} = m$ implies that agent a_i has queried m times object o_j across w.

2.2 Groups

We now define a structure called a *social group*, which relates agents that have carried out a query over the same collection of objects for a given time window.

Definition 1 (Social Group). *Let w be a window, with agents, $\mathrm{agt}(w)$, and objects, $\mathrm{obj}(w)$. Then, the tuple*

$$\langle w, A \subseteq \mathrm{agt}(w), O \subseteq \mathrm{obj}(w)\rangle$$

written $g^w(A, O)$ for short, forms a social group *of* size $|O|$*, and* weight $|A|$*, iff every agent in A has queried all objects in O, in symbols:*

$$\forall x \in A.\ \forall y \in O.\ \mathrm{qry}^w(x, y)$$

Notice that, in particular, for a given group $g^w(A, O)$, qry^w is the Cartesian product of $A \times O$.

Definition 2 (Size-/Weight-Maximal Group). *Let $w \in \mathbb{W}$ be a window, and let G^w denote all the existing groups in w. Then, a group $g^w(A, O) \in G^w$ is called a* size-maximal group *of G^w if there does not exist $g^w(A', O') \in G^w$ such that $|O| < |O'|$.*

Similarly, a group $g^w(A, O) \in G^w$ is called a weight-maximal group *of G^w if there does not exist $g^w(A', O') \in G^w$ such that $|A| < |A'|$.*

Clearly, we can build a poset out of G^w, using a lexicographical order, $\prec$, which combines the two previous posets, namely: size, and weight, in that order.

Definition 3 ($\preceq$, $\prec$, Maximal Group). *Define $\preceq$ and $\prec$ as follows:*

— $\langle a, b\rangle \preceq \langle c, d\rangle$ *iff* $a < b$, *or (*$a = b$ *and* $c \leq d$*), and*

— $s \prec s'$ *if* $s \preceq s'$ *and* $s \neq s'$.

Then, let $w \in \mathbb{W}$ be a window, and let G^w denote all the existing groups in w. Then, a group $g^w(A, O) \in G^w$ is called a maximal group *of G^w if there does not exist $g^w(A', O') \in G^w$ such that $\langle |O|, |A|\rangle \prec \langle |O'|, |A'|\rangle$.*

2.3 Queried Objects and Queried by Individuals

We now define symbols that collect information about individuals' activities.

Definition 4 (Agent Cover, Object Attraction). *Let $w \in \mathbb{W}$ be a window, with agents $\mathrm{agt}(w) = \{a_1, a_2, \ldots\} \subset \mathbb{A}$, and objects $\mathrm{obj}(w) = \{o_1, o_2, \ldots\} \subset \mathbb{O}$. Then, the* cover *of an agent a_i, with respect to w, written $q_i(w)$, is a list, just as w, except that it contains all the objects queried by agent a_i, following w's order of appearance:*

$$q_i(w) = \langle o_j, o_k, \ldots\rangle \text{ whenever, }$$
$$w = \langle \ldots, (a_i, o_j), \ldots, (a_i, o_k), \ldots\rangle$$

Likewise, the attraction *of an object o_j, with respect to w, written $\mathrm{trk}_j(w)$, is the list of all agents that have queried o_j:*

$$\mathrm{trk}_j(w) = \langle a_i, a_k, \ldots\rangle \text{ whenever, }$$
$$w = \langle \ldots, (a_i, o_j), \ldots, (a_k, o_j), \ldots\rangle$$

2.4 "Real" world examples of SGC

Example 1. A team of market researchers is interested in identifying groups of clients of an online bookshop with common book interests. Since the set of client book purchases is rather huge, the team decides to segment the purchase record history using a *sliding window* approach, therefore fixing the window size (for instance, the number of purchases) and the window step (for instance, the number of purchases the window is to be slid for the next group analysis.) For the sake of simplicity, suppose the research team has fixed the window size to 9 purchases, and that, currently, for some window w has got the following observation:

$$w = \langle (c_1, b_1), (c_1, b_2), (c_1, b_3), (c_2, b_4),$$
$$(c_2, b_3), (c_2, b_2), (c_3, b_1), (c_3, b_4), (c_2, b_1)\rangle$$

where we use c, and b to denote a client and a book, respectively. Then, clients and books form the sets $\mathrm{agt}(w) = \{c_1, c_2, c_3\}$, and $\mathrm{obj}(w) = \{b_1, b_2, b_3, b_4\}$. The client purchase record, as expressed by w, is then used to respectively compute the covers of agents c_1, c_2, and c_3: $q_1 = \langle b_1, b_2, b_3\rangle$, $q_2 = \langle b_4, b_3, b_2, b_1\rangle$, and $q_3 = \langle b_1, b_4\rangle$.

Clients c_1 and c_2 form a group of size and weight two, since both have purchased books b_2 and b_3, in symbols: $g(\{c_1, c_2\}, \{b_2, b_3\})$. Notice that clients

DOI: 10.13053/CyS-17-4-2013-003

c_2 and c_3 also form a group of the same measures, this time, though, given by $g(\{c_2, c_3\}, \{b_1, b_4\})$, as they have in common the purchases b_1 and b_4. With this information, the market research team might issue a campaign, offering, *e.g.*, for sale the books that clients do not have in common.

Example 2. Suppose now that we are interested in studying how a collection of concepts, $c_1, c_2, \ldots$, are referred to in a few wikipedia articles, $a_1, a_2, \ldots$. Also, suppose, that at some time, we have analyzed these articles and taken the following observation:

$$w = \langle (c_1, a_1), (c_1, a_2), (c_1, a_3), (c_2, a_4), \\ (c_2, a_3), (c_2, a_2), (c_3, a_1), (c_3, a_4), (c_2, a_1)\rangle$$

Then, $\mathrm{agt} = \{c_1, c_2, c_3\}$ and $\mathrm{obj} = \{a_1, a_2, a_3, a_4\}$, with agent covers $q_1 = \langle a_1, a_2, a_3, a_4\rangle$, $q_2 = \langle a_4, a_2, a_1\rangle$, and $q_3 = \langle a_2, a_3\rangle$. Notice, however, that in this case q_i denotes the articles where concept c_i appears.

Again, we find two groups. One, $g(\{c_1, c_2\}, \{a_1, a_2, a_4\})$ conveys, that concepts c_1 and c_2 appear in three articles: a_1 and a_2, and a_4; and the other, $g(\{c_2, c_3\}, \{a_2, a_3\})$, that concepts c_2 and c_3 appear in articles a_2 and a_3.

Further examples of SGC are given by simply substituting agents, objects and the action on a particular scenario. For example, students and books correspond to agents and objects, respectively, while the action could be buying. As another example, we can consider users as agents, webpages as objects, and visiting a web page using a browser as the action.

2.5 Problem Statement

Take a window, $w \in \mathbb{W}$, and two positive integers, z and t. Then, the problem that asks for *all* groups, g^w, in w, having size k or less and weight t or less is, clearly, provably intractable. This is because it is easy to construct instances of the problem where exponentially many groups are smaller than or equal to the given bound; this way, no polynomial time algorithm could possibly list them all. As pointed out by Garey and Jhonson [5], this problem formulation might not be realistic, in that it involves more information than one could hope to use. This remains true for our problem unless we are trying to compute a maximal group, or to compare two or more populations in terms of their activity, as is the case for the detection of a denial of service attack to a DNS server (see, e.g., [1]), and many other cases.

Accordingly, we shall cast the calculation of social groups as a decision problem, having two possible solutions, namely: “yes” or “no”, in a way that it becomes of practical interest. In addition, this casting is necessary as we shall be constructing a Karp reduction from a well-known decision problem to ours, when proving NP-completeness of social group calculation.

The decision version of the social group calculation problem can be defined as shown below.

Definition 5 (Social Group Calculation, SGC).

INSTANCE: A window $w \in \mathbb{W}$*, a finite set* $\mathrm{obj} = \{o_1, o_2, o_3, ..., o_n\} \subset \mathbb{O}$ *of objects, a finite set* $\mathrm{agt} = \{a_1, a_2, a_3, ..., a_m\} \subset \mathbb{A}$ *of agents, a finite set* $Qy = \{q_1, q_2, q_3, ..., q_m\}$ *of agent covers, one for each agent, a positive integer,* $z > 0$ *and, a positive integer,* $t > 0$*.*

QUESTION: Is there a group of size lesser than or equal to z *and weight* t*?*

Example 3. Let $w = \langle (a_1, o_1), (a_1, o_2), (a_1, o_3), (a_2, o_4), (a_2, o_3), (a_2, o_2), (a_3, o_1), (a_3, o_4)\rangle$, $\mathrm{agt} = \{a_1, a_2, a_3\}$, $\mathrm{obj} = \{o_1, o_2, o_3, o_4\}$, $Qy = \{q_1, q_2, q_3\}$ with agent covers $q_1 = \langle o_1, o_2, o_3\rangle$, $q_2 = \langle o_4, o_3, o_2\rangle$, $q_3 = \langle o_1, o_4\rangle$. Together, they constitute a yes-instance of SGC, with $z = 2$ and $t = 2$, witnessed by the group $g(\{a_1, a_2\}, \{o_2, o_3\})$.

Remark Determining whether or not there exists a group of size z and weight t in a given window is in the class P, under the proviso that we do not bother ourselves to produce a witness (see Example 3). To see this, simply notice that social group information can be obtained by looking into the matrix product[1] $Q \times Q^T$ of the query matrix, Q, and its associated transpose, Q^T. Actually, from $Q \times Q^T$, it is possible to determine how many groups Q has, together with their corresponding sizes and weights. Inspecting this matrix product entirely is not complex and is further alleviated by both facts: that $Q \times Q^T$

[1] We are grateful and indebted to Johan Van Horebeek (from CIMAT) for making this observation.

DOI: 10.13053/CyS-17-4-2013-003

is symmetric and that the main diagonal carries other information (namely, the top active agents). However, determining a group of a given size z and weight t is provably intractable, that is, producing the witness $g(\mathrm{agt}, \mathrm{obj})$ cannot be carried out in polynomial time.

3 SGC is NP-complete

The classic paper of Karp [5, 7] was a significant breakthrough towards the understanding of complex problems. Karp proves the NP-completeness of several well-known problems using a mapping procedure later known as a *Karp reduction*. After that, several works on proving NP-completeness appeared. As an example during the last 12 years works like [3, 15, 4, 13] showed the validity of using Karp's idea.

Karp showed that for proving the NP-completeness of an unknown problem, π', one may follow a five-step procedure. In the first step, we select a problem, π, that has been proven to be NP-complete (in principle, any NP-complete problem would do, but a careful selection makes it easier to find the proof of the third step, see below). Next, in the second step, we prove that π' is in NP. Then, in the third step, we show how to transform π to π'; this step is typically known as the *reduction* and denoted by $\pi \leq_p \pi'$. In the fourth step, we show that the reduction can be carried out in polynomial time. Finally, in the fifth step, we prove that whenever there is an answer in π, then there also is an answer in π', and vice versa.

As expected, we shall follow this proof procedure for establishing the main results of this paper.

3.1 Longest Common Subsequence Problem

For our first reduction, we have chosen the Longest Common Subsequence (LCS) problem for the Karp reduction: $LCS \leq_p SGC$. LCS is NP-complete and well-known [9], for it arises in many contexts [2], such as bioinformatics [8] or file comparison (c.f. the UNIX `diff` command.) Let $\mathrm{length}(\cdot)$ be a polymorphic function, which has the natural interpretation, returning the number of elements of its input argument; then LCS is defined as follows:

Definition 6 (Longest Common Subsequence).

INSTANCE: A finite alphabet Σ, a set R of strings from Σ^*, and a positive integer k.

QUESTION: Is there a string $s' \in \Sigma^*$, with $\mathrm{length}(s') \geq k$, such that s is a subsequence of each $s' \in R$?

Example 4. Let $\Sigma = \{a, b, c, d\}$, $R = \{s_1, s_2\}$, with strings $s_1 = abcd$, $s_2 = dadbcaa$,[2] and let $k = 3$. Together, they constitute a yes-instance of LCS, with $k = 3$, witnessed by $s' = abc$, the longest common subsequence for the strings s_1 and s_2.

3.2 SGC NP-Completeness

Theorem 1: SGC is NP-complete.

Proof: Having fixed π to be LCS, we then prove that SGC is in NP. To see this, notice that any instance of SGC can be solved using a Non-Deterministic Turing Machine (NDTM), which, upon halting, provides a witness, $g(A, O)$. Verifying that $g(A, O)$ truly is a witness can be certainly carried out in polynomial time, as it consists of checking that every agent in the set A queries all objects in the set O.

We now proceed to produce the reduction, LCS $\leq_p$ SGC. We use $I, J, \ldots$ stand for indexing sets, and write $\widetilde{\Sigma_I} = \{\ell_i : \ell \in \Sigma, i \in I\}$ to denote the set of symbols in Σ, indexed by I. Let $s|_p$ denote the element e at position $p \in \{1, \ldots, \mathrm{length}(s)\}$ of s, either a list or a string. Now, let Σ be a finite alphabet, R a set of strings from Σ^*, and let k be an integer, such that they all constitute an instance of the LCS problem. Our reduction maps the parameters of an LCS instance to an SGC instance, as follows:

1. For each alphabet symbol, $\ell_i \in \widetilde{\Sigma_I}$, create a unique object, $o_i \in \widetilde{\mathrm{obj}_I}$, and then build the bijective function, $\mapsto_I$, which associates every symbol in Σ with an object in obj: $\widetilde{\Sigma_I} \mapsto_I \widetilde{\mathrm{obj}_I}$.
2. For each string, $s_j \in \widetilde{R_J}$, create a unique agent, $a_j \in \widetilde{\mathrm{agt}_J}$, and then build the bijective function $\mapsto_J$, such that $\widetilde{R_J} \mapsto_J \widetilde{\mathrm{agt}_J}$.

[2] Following standard notation, we use juxtaposition to denote the string constructor function.

DOI: 10.13053/CyS-17-4-2013-003

3. For each string, $s_j \in \widetilde{R_J}$, build the associated cover of the agent a_j, q_j, such that $\mathrm{length}(s_j) = \mathrm{length}(q_j)$, and such that, for all $p \in \{1, \ldots, \mathrm{length}(s_j)\}$, $s_j|_p \mapsto_J q_j|_p$; in this way, we also build Qy.

4. For each agent cover, $q_j = \langle o_i, o_k, \ldots \rangle$, build the expected agent (sub-)window: $w_j = \langle (a_j, o_i), (a_j, o_k), \ldots \rangle$. Next, build w simply by concatenating all agent windows.

5. Finally, set $z = k$ and $t = |R|$.

Example 5. Consider an instance of LCS, where $\Sigma = \{a, b, c, d, e\}$, $R = \{s_1, s_2\}$, $s_1 = abbdc, s_2 = abdeb$, and $k = 3$. Following our *Karp reduction*, we generate the next instance of SGC:

1. $\mathrm{obj} = \{o_1, o_2, o_3, o_4, o_5\}$;
2. $\mathrm{agt} = \{a_1, a_2\}$;
3. $q_1 = \langle o_1, o_2, o_2, o_4, o_3 \rangle$, $q_2 = \langle o_1, o_2, o_4, o_5, o_2 \rangle$, and $Qy = \{q_1, q_2\}$;
4. $w = \langle (a_1, o_1), (a_1, o_2), (a_1, o_2), (a_1, o_4), (a_1, o_3),$
   $(a_2, o_1), (a_2, o_2), (a_2, o_4), (a_2, o_5), (a_2, o_2) \rangle$;
5. $z = 3$, and $t = 2$.

Our reduction can be carried out in polynomial time; indeed, it is clearly linear in the number of steps plus the number and length of each string.

Now, the only step left is to prove that, for any LCS instance, there exists a common subsequence with $\mathrm{length}(s') \geq k$, if and only if there also is a group of size $z = k$ and weight $t = |R|$ in the generated SGC instance.

$(\Longrightarrow)$ Take again an instance of LCS, with Σ, a finite alphabet, R, a set of strings from Σ^*, and k, a positive integer. Also, take this instance to be positive, with witness s. To transform s into a SGC witness, g, carry out our reduction, and then proceed as follows:

1. Set
   $A = \mathrm{agt}$, and
   $O = \{o \mid s|_p \mapsto_J o, p \in \{1, \ldots, \mathrm{length}(s)\}\}$.

2. Finally, set $g(A, O)$, which stands for a group of size $|O|$ and length $|A|$.

Example 6. Consider again Example 5. The LCS instance is actually positive: $s = abd$ is a witness longest common subsequence, with $k = 3$. Carrying out the previous procedure, we end up with a SGC instance which is also positive, witnessed by $g(\{a_1, a_2\}, \{o_1, o_2, o_4\})$, with size $z = 3$ and weight $t = 2$.

$(\Longleftarrow)$ We first prove that finding a group in the generated SGC instance implies finding a common subsequence for the LCS instance. Take the generated instance of SGC, with agt, obj, Qy, w, z and t. Then, to find a group, if any, proceed as follows:

1. Set $A = \mathrm{agt}$. Find $O \subseteq \mathrm{obj}$, such that $|O| = z$, and such that $g(A, O)$ forms a group of size $z = |O|$, and weigth $t = |A|$. If there does not exist one such a group, halt with failure (see below); otherwise convert O into a list, O', and continue.

2. Using the inverse of $\mapsto_J$,[3] transform O' into the string $\widehat{O'}$.

3. Transform $\widehat{O'}$, imposing an ordering, so that it is a subsequence of every $\widehat{s_j}$ $(j \in J)$.

4. Call the transformed string, s, actually the witness of the LCS instance; that is, s is a longest common subsequence of every string in R.

Example 7. Let us go back again to the generated SGC instance of Example 5, and assume we have now set $k = 2$. Then, using the previous procedure, we could have picked up the group $g(\{a_1, a_2\}, \{o_2, o_4\})$ of size $z = 2$ and weight $t = 2$. This, in turn, yields the LCS witness $s' = bd$ for $k = 2$.

Consider now the case where we have $A \subseteq \mathrm{agt}$ and $O \subseteq \mathrm{obj}$, such that A and O are not a group of size $|O|$ and weight $|A|$. Then, it only remains to prove that O cannot be transformed into a common subsequence, for a positive answer of the LCS problem instance. To see this, notice that, since A and O are not a group, not all the elements of O appear in the associated agent covers, and, thus,

[3] Recall that $\mapsto_I$ and $\mapsto_J$ are bijective, so it is guaranteed that they both have an inverse.

DOI: 10.13053/CyS-17-4-2013-003

O cannot be transformed into a string that is a subsequence of every string in R, out of which we have computed the agent covers.

Example 8. Consider Example 5, let $A = \{a_1, a_2\}$ and $O = \{o_4, o_5\}$. Together, these sets are not a group of size $z = 2$ and weight $t = 2$. Transforming this SGC answer to a LCS witness gives us $s = de$, which can easily be seen not to be a valid common subsequence with $\mathrm{length}(s) \geq k = 2$.

□

4 2-SGC is NP-complete

We now consider the scenario, where, given an observation window, we would like to know if every agent belongs to some group, not necessarily the same one, of a size and a weight at least equal to 2. This special case of the SGC model, which we call *2-SGC*, is still NP-complete. Our result follows by means of a Karp reduction from the well-known Hitting Set problem and relies on a graphical representation of 2-SGC.

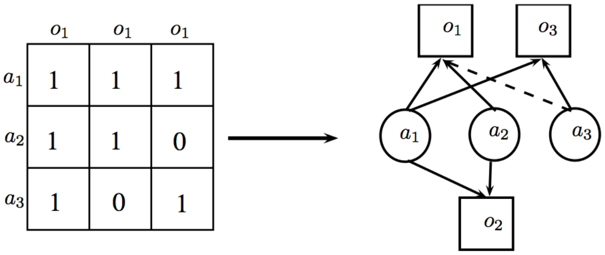
4.1 2-SGC Problem Statement

A query matrix Q^w, associated with window $w \in \mathbb{W}$, can be easily turned into an incidence matrix, by setting $Q^w_{i,j} = 1$, if agent a_i has queried object o_j along w, and $Q^w_{i,j} = 0$, otherwise. An incidence matrix in turn gives rise to a graph, which we call *connectivity graph*, $G = (V, E)$, which is such that $V = \mathrm{obj} \cup \mathrm{agt}$, and $E = \{(a_i, o_j) \mid Q^w_{i,j} = 1\}$ (see Fig. 1). We formalize 2-SGC as follows (as before, when understood from the context, we shall refrain ourselves from explicitly noting the window, $w \in \mathbb{W}$, upon which observations are made):

Definition 7 (2-SGC).

> *INSTANCE: A connectivity graph $G = (V, E)$ with $V = \mathrm{obj} \cup \mathrm{agt}$ and $E = \{(a_i, o_j) \mid Q_{i,j} = 1\}$, for a given Q.*
>
> *QUESTION: Is there a 2-SGC?,* i.e., *does every agent, $a_i \in \mathrm{agt}$, belong to a group, involving objects $o_j \in \mathrm{obj}$, of a size and a weigth at least equal to two?*

Example 9. Fig. 1 portrays an example connectivity graph, where $G = (V, E)$, with $V = \mathrm{obj} \cup \mathrm{agt}$, $\mathrm{obj} = \{o_1, o_2, o_3\}$ and $\mathrm{agt} = \{a_1, a_2, a_3\}$, and where $E = \{(a_1, o_1, (a_1, o_2\}, (a_1, o_3), (a_2, o_2),$ $(a_2, o_1), (a_3, o_1), (a_3, o_3)\}$.
It actually is a yes-instance of 2-SGC, witnessed by $g(\{a_1, a_3\}, \{o_1, o_3\})$ and $g(\{a_1, a_2\}, \{o_1, o_2\})$. Notice that if we removed the dotted line, the instance would no longer be of type yes.



$$\begin{array}{c|c|c|c|}
 & o_1 & o_1 & o_1 \\ \hline
a_1 & 1 & 1 & 1 \\ \hline
a_2 & 1 & 1 & 0 \\ \hline
a_3 & 1 & 0 & 1 \\ \hline
\end{array}$$

Fig. 1. Incidence matrix Q and associated connectivity graph; as it is standard in the literature (resource usage), agents are denoted with circles and objects with squares

4.2 Hitting Set

For proving 2-SGC NP-completeness, we have chosen Hitting Set (HS) [7, 12]:

Definition 8 (Hitting Set, HS).

> *INSTANCE: A finite set S, a collection $\mathcal{C}$ of subsets of S, and a positive integer $k \leq |S|$.*
>
> *QUESTION: Is there a hitting set?* i.e.*, is there a subset $S' \subseteq S$ such that S' has at least one element of each subset of $\mathcal{C}$ with $|S'| \leq k$?*

Example 10. Take $S = \{s_1, s_2, s_3, s_4\}$, $\mathcal{C} = \{C_1, C_2, C_3\}$, $C_1 = \{s_1, s_2, s_3\}$, $C_2 = \{s_2, s_4, s_3\}$, and $C_3 = \{s_1, s_2\}$. Together, they constitute a yes-instance of HS, with $k = 3$, witnessed by $S' = \{s_2, s_3, s_4\}$.

DOI: 10.13053/CyS-17-4-2013-003

4.3 2-SGC NP-Completeness

Theorem 2: 2-SGC is NP-complete.

Proof: Clearly, an NDTM can be used to solve any instance of the 2-SGC problem, yielding a collection of groups. Verifying the witness 2-SGC amounts to verifying that each agent belongs to a group of a size and a weight at least equal to two. As argued in Section 3.2 group verification can be carried out in polynomial time. Thus, 2-SGC is in NP.

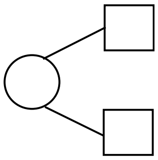


Fig. 2. A partially social cell

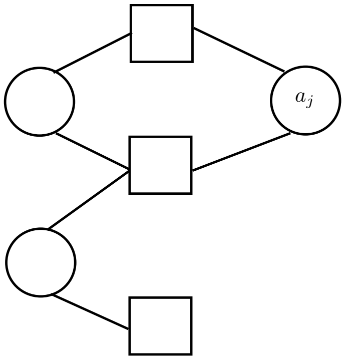


Fig. 3. New agent chained to the component

We shall now provide a Karp reduction: $\mathrm{HS} \leq_p 2-\mathrm{SGC}$. Consider an instance of HS, with S, a finite set, $\mathcal{C}$, a collection of subsets of S, and $k \geq |S|$, an integer. We shall index S and $\mathcal{C}$ using I and J, respectively. Then, for each element $s_i \in \widetilde{S_I}$, we construct a social group as follows:

1. Add what we call a *partially social cell*, consisting of an agent querying for two objects (see Fig. 2).
2. If there is a set C_j containing s_i, complete the most recently added social cell, adding an agent labeled a_j (see Fig. 4), thus forming a group of size and weight equal to two. Otherwise, remove the partially social cell, skip the rest of these steps, going back to step 1 and continuing with the next $s_i \in \widetilde{S_I}$. Henceforth, we shall call a chain of social cells a *social component*.
3. Then, add a partially social cell to the current social component in such a way that the newly added agent is connected with of one of the resources (see Fig. 3).
4. If there is another C_j containing s_i go back to step 2; otherwise, box the social component, forming a subgraph, and label this graph G_i, using the same index as the current s_i (see Fig. 5). Then, go back to step 1, continuing with the next $s_i \in \widetilde{S_I}$.

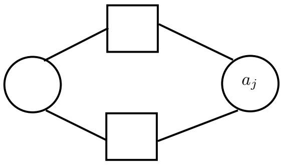


Fig. 4. A social cell

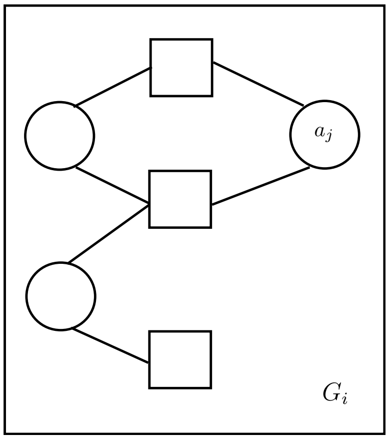


Fig. 5. Boxed component, G_i, comprising agent a_j

Notice that in our reduction, the objects that are introduced for the 2-SGC are dummy. An example transformation of HS to 2-SGC is given below.

Example 11. Let $S = \{s_1, s_2, s_3, s_4\}$, $\mathcal{C} = \{C_1, C_2, C_3\}$, with $C_1 = \{s_1, s_2, s_3\}$, $C_2 = \{s_2, s_4\}$, $C_3 = \{s_1, s_2, s_4\}$, and $k = 2$ be an instance of the HS problem. After applying the previous transformation, we get the connectivity graph shown in Fig. 6.

DOI: 10.13053/CyS-17-4-2013-003

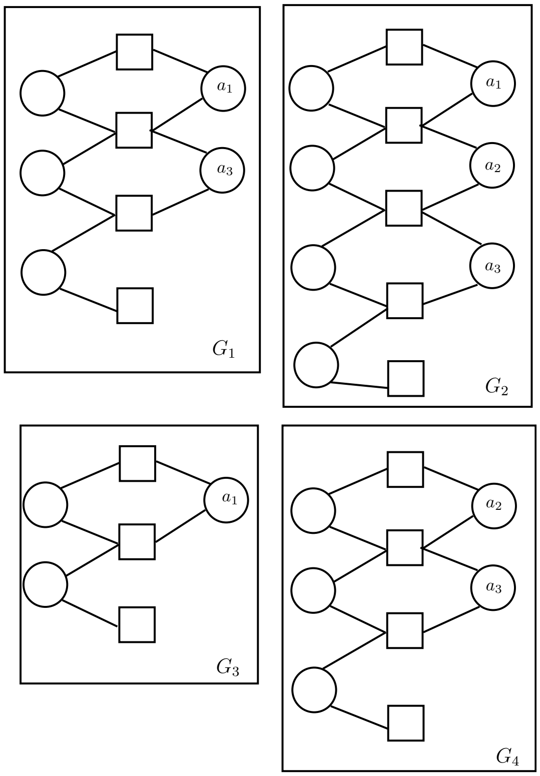


Fig. 6. The 2-SGC instance that results from applying our reduction to the HS instance $S = \{s_1, s_2, s_3, s_4\}$, $\mathcal{C} = \{C_1, C_2, C_3\}$, with $C_1 = \{s_1, s_2, s_3\}$, $C_2 = \{s_2, s_4\}$, and $C_3 = \{s_1, s_2, s_4\}$

Notice that our reduction can be performed in polynomial time, since the construction of the graph can be done in linear time on the cardinality of S. So the only step left is to prove that a hitting set with $|S'| \leq k$ exists if and only if 2-SGC holds in the output connectivity graph.

($\Longrightarrow$) Take a yes instance of HS, with S, a finite set, $\mathcal{C}$ a collection of subsets of S, and $k \leq |S|$, an integer, such that S' is the corresponding witness, with $|S'| \leq k$. Then, collect together in a single connectivity graph all the box components, labeled G_i, for each i such that $s_i \in S'$: $G = \bigcup_{s_i \in S'} G_i$. Since, by construction, every agent a_j in a boxed component G_i is part of a social cell, it follows that every agent belongs to a group that is of a size and weight equal to two; therefore, we have yielded a yes 2-SGC instance.

($\Longrightarrow$) Finally, to transform a generated 2-SGC answer into one of HS, simply select at least k boxed components, such that they contain all the agents appearing in the 2-SGC instance. To transform the 2-SGC answer to a HS answer, construct the HS witness using the corresponding s_i using the label of the selected box component G_i.

Example 12. Consider again the HS instance of Example 11, together with the associated output connectivity graph 2-SGC instance, yielded by our reduction, and shown in Fig. 6. The set $S' = \{s_1, s_4\}$ is hitting, for $|S'| \leq k = 2$, since it contains at least one element of each $C_j \in \mathcal{C}$. We map S' to a 2-SGC witness, selecting for each $s_i \in S'$ the corresponding box component labeled G_i, G_1 and G_4, and forming the corresponding connectivity graph $G = G_1 \cup G_4$ (see Fig. 7), in which clearly every agent a_i belongs to a group of size and weight equal to two.

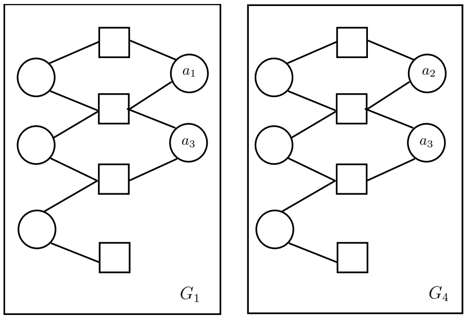


Fig. 7. 2-SGC made out of social components G_1 and G_4, taken from the graph shown in Fig. 6

By contrast, notice that if we selected an invalid witness from the ouput connectivity grap shown in Fig. 11, say $G = G_3 \cup G_4$ (see Fig. 8), we would produce $S' = \{s_3, s_4\}$, which is also invalid as a witness for the associated HS problem. Thus, there is a hitting set, with $|S'| \leq k$, if and only if there is a 2-SGC in the connectivity graph output by our reduction.

With this, we have completed the fifth step of the Karp reduction and also our proof of 2-SGC NP-completeness. □

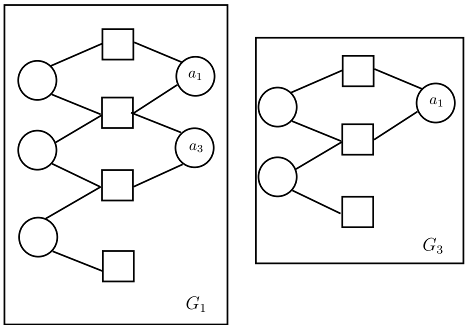


Fig. 8. 2-SGC made out of social components G_1 and G_3, taken from the graph shown in Fig. 6

5 Conclusion and Future Work

We introduced the Social Group Commonality (SGC) problem. Basically, SGC queries the commonality among the objects upon which a collection of agents has performed an action. SGC has several applications including modeling of DNS servers for the detection of Distributed Denial of Service (DDoS) attacks [1], multidimensional clustering for marketing purposes, modeling of system users, wikification, etc. SGC is simple to formulate and interestingly hard.

The SGC problem and the special case 2-SGC are NP-complete. The theorems presented in this work give some insight into the difficulty of grouping agents on the basis of the actions performed on a set of objects. It also proves the inexistence of a polynomial algorithm capable of computing a group in SGC (Theorem 1) and determining if every agent belongs to some group, not necessarily the same one, in 2-SGC (Theorem 2), unless P=NP. Moreover, if we look for a maximal group, SGC might as well belong to the NP-hard class since the NDTM needs to compute all groups to verify the answer. Thus, ongoing work includes proving NP-Hardness of SGC, using a Turing reduction.

The best algorithm to solve SGC remains open. A naïve algorithm has an exponential number of operations. This algorithm computes combinations of objects making polynomially intractable to find every group in a window. Algorithms to solve, correctly and completely, SGC can use the idea of the absence of some objects combinations. For example, a group of Facebook users, John and George with common friends Laura, Lisa and Paul, exists if and only if John and George are friends of Laura and Paul, if John and George are friends of Laura and Lisa, and if John and George have the common friends Lisa and Paul. From this idea we can drastically reduce the search space when looking for all groups in a window.

We are currently working on finding the phase transition of SGC. Roughly speaking, phase transition helps us to identify instances of NP-complete problems solvable in polynomial time. Current experimental data indicate that the complexity of an instance is given by the size and weight of the groups in a window.

The results presented in this work provide a better understanding of this (to the authors' knowledge) so far ignored problem. The results also motivate research in an area where the best algorithm, heuristics, and applications for SGC are the main topics to be covered.

Acknowledgement

This paper has largely benefited from numerous discussions with Luis Ángel Trejo-Rodríguez. We thank to the members of the NetSec group at Tecnológico de Monterrey, Estado de México, for their constructive comments on an earlier version of this paper. The first author was supported by CONACYT student scholarship 45904, while the second author was in part supported by CONACyT grant 105698.

References

1. **Alonso, R., Vazquez, J., Trejo, L., Monroy, R., & Sanchez, E.** (**2009**). How social networks can help detecting ddos attacks on dns servers. In *Artificial Intelligence and Applications, Complementary Proceedings of MICAI 2009, 8th Mexican International Conference on Artificial Intelligence*. Sociedad Mexicana de Inteligencia Artificial, Guanajuato, Mexico. ISBN 978-607-95367-1-8, 261–270.

2. **Bergroth, L., Hakonen, H., & Raita, T.** (**2000**). A survey of longest common subsequence algorithms. In *Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00)*, SPIRE '00. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-0746-8, 39–48.

3. **Díaz, J., Pottonen, O., Serna, M., & van Leeuwen, E. J.** (**2012**). On the complexity of metric dimension. In *Proceedings of the 20th Annual European conference on Algorithms*, ESA'12. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-642-33089-6, 419–430. doi:10.1007/978-3-642-33090-2_37.

4. **Dyer, M. & Greenhill, C.** (**2000**). The complexity of counting graph homomorphisms. In *Proceedings of the ninth international conference on on Random structures and algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 260–289.

5. **Garey, M. R. & Johnson, D. S.** (**1990**). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA. ISBN 0716710455.

6. **Horst, H.** (**2005**). Combining rdf and part of owl with rules: Semantics, decidability, complexity. In **Gil, Y., Motta, E., Benjamins, V., & Musen, M.**, editors, *The Semantic Web – ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-29754-3, 668–684. doi:10.1007/11574620_ 48.

7. **Karp, R.** (**1972**). Reducibility among combinatorial problems. In **Miller, R. & Thatcher, J.**, editors, *Complexity of Computer Computations*. Plenum Press, 85–103.

8. **Liu, W., Chen, L., & Zou, L.** (**2007**). A parallel lcs algorithm for biosequences alignment. In *Proceedings of the 2nd international conference on Scalable information systems*, InfoScale '07. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium. ISBN 978-1-59593-757-5, 83:1–83:8.

9. **Maier, D.** (**1978**). The complexity of some problems on subsequences and supersequences. *J. ACM*, 25(2), 322–336. ISSN 0004-5411. doi:10.1145/322063.322075.

10. **Mihalcea, R. & Csomai, A.** (**2007**). Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07. ACM, New York, NY, USA. ISBN 978-1-59593-803-9, 233–242. doi:10.1145/1321440.1321475.

11. **Milne, D. & Witten, I. H.** (**2008**). Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08. ACM, New York, NY, USA. ISBN 978-1-59593-991-3, 509–518. doi:10.1145/1458082.1458150.

12. **Paschos, V. T.** (**1997**). A survey of approximately optimal solutions to some covering and packing problems. *ACM Comput. Surv.*, 29(2), 171–209. ISSN 0360-0300. doi:10.1145/254180.254190.

13. **Schmitt, M. & Martignon, L.** (**2006**). On the complexity of learning lexicographic strategies. *J. Mach. Learn. Res.*, 7, 55–83. ISSN 1532-4435.

14. **Witten, I. H.** (**2010**). Semantic document processing using wikipedia as a knowledge base. In *Proceedings of the Focused retrieval and evaluation, and 8th international conference on Initiative for the evaluation of XML retrieval*, INEX'09. Springer-Verlag, Berlin, Heidelberg. ISBN 3-642-14555-8, 978-3-642-14555-1, 3–3.

15. **Xu, D., Chen, Y., Xiong, Y., Qiao, C., & He, X.** (**2006**). On the complexity of and algorithms for finding the shortest path with a disjoint counterpart. *IEEE/ACM Trans. Netw.*, 14(1), 147–158. ISSN 1063-6692. doi:10.1109/TNET.2005.863451.

Roberto Alonso is a Ph.D. student in Computer Science at Tecnológico de Monterrey, Campus Estado de México (CEM). His thesis work is related to the detection of security anomalies on DNS servers. Since 2012, he is a part-time lecturer at Tecnológico de Monterrey, CEM. His research interests include complexity theory, particularly, proving NP-completeness of problems, pattern recognition, data mining and computer security.

Raúl Monroy obtained a Ph.D. in Artificial Intelligence in 1998 from Edinburgh University, under the supervision of Prof. Alan Bundy. He is a (full) Professor in Computing at Tecnológico de Monterrey, Campus Estado de México. Since 1998, he is a member of CONACyT's National Research System, currently rank 2. Dr. Monroy's research interests are in discovery and application of general search control strategies for uncovering and correcting errors in either a system or its specification, detecting anomalies endangering information security and planning robot motion.

Article received on 19/12/2012; accepted on 15/01/2013.

DOI: 10.13053/CyS-17-4-2013-003