Group Activity Selection with Few Agent Types

Robert Ganian
TU Wien, Vienna, Austria
ganian@ac.tuwien.ac.at

Sebastian Ordyniak
University of Sheffield, Sheffield, UK
sordyniak@gmail.com

C. S. Rahul
University of Warsaw, Warsaw, Poland
rahulcsbn@gmail.com

Abstract

The Group Activity Selection Problem (GASP) models situations where a group of agents needs to be distributed to a set of activities while taking into account preferences of the agents w.r.t. individual activities and activity sizes. The problem, along with its well-known variants sGASP and gGASP, has previously been studied in the parameterized complexity setting with various parameterizations, such as *number of agents*, *number of activities* and *solution size*. However, the complexity of the problem parameterized by the *number of types of agents*, a natural parameter proposed already in the first paper that introduced GASP, has so far remained unexplored.

In this paper we establish the complexity map for GASP, sGASP and gGASP when the number of types of agents is the parameter. Our positive results, consisting of one fixed-parameter algorithm and one XP algorithm, rely on a combination of novel Subset Sum machinery (which may be of general interest) and identifying certain *compression steps* which allow us to focus on solutions which are "acyclic". These algorithms are complemented by matching lower bounds, which among others close a gap to a recently obtained tractability result of Gupta, Roy, Saurabh and Zehavi (2017). In this direction, the techniques used to establish W[1]-hardness of sGASP are of particular interest: as an intermediate step, we use Sidon sequences to show the W[1]-hardness of a highly restricted variant of multi-dimensional Subset Sum, which may find applications in other settings as well.

2012 ACM Subject Classification Theory of computation → Exact and approximate computation of equilibria; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases group activity selection problem, parameterized complexity analysis, multi-agent systems

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.48

Related Version A full version of the paper is available at `https://arxiv.org/abs/1808.06954`.

Acknowledgements Robert Ganian acknowledges support by the Austrian Science Fund (FWF, Project P31336), and is also affiliated with FI MUNI, Czech Republic.

1 Introduction

In this paper we consider the Group Activity Selection Problem (Gasp) together with its two most prominent variants, the Simple Group Activity Selection Problem (sGasp), and the Group Activity Selection Problem with Graph Structure (gGasp) [6, 18]. Since their introduction, these problems have received considerable attention, notably in venues dedicated to multi-agent systems and game theory [3, 4, 5, 19, 14, 15]. In Gasp one is given a set of agents, a set of activities, and a set of preferences for each agent in the form of a complete transitive relation (also called the *preference list*) over the set of pairs consisting of an activity a and a number s, expressing the willingness of the agent to participate in the activity a if it has s participants. The aim is to find a "good" assignment from agents to activities subject to certain *rationality* and *stability* conditions. Specifically, an assignment is individually rational if agents that are assigned to an activity prefer this outcome over not being assigned to any activity, and an assignment is (Nash) stable if every agent prefers its current assignment over moving to any other activity. In this way GASP captures a wide range of real-life situations such as event organization and work delegation.

sGasp is a simplified variant of Gasp where the preferences of agents are expressed in terms of *approval sets* containing (activity, size) pairs instead of preference lists. In essence sGasp is Gasp where each preference list has only two equivalence classes: the class of the approved (activity, size) pairs (which contains all pairs that are preferred over not being assigned to any activity), and the class of disapproved (activity, size) pairs (all possible remaining pairs). On the other hand, gGasp is a generalization of Gasp where one is additionally given an undirected graph (network) on the set of all agents that can be employed to model for instance acquaintanceship or physical distance between agents. Crucially, in gGasp one only considers assignments for which the subnetwork induced by all agents assigned to some activity is connected. Note that if the network forms a complete graph, then gGasp is equivalent to the underlying Gasp instance.

Related Work. sGasp, Gasp, and gGasp, are known to be NP-complete even in very restricted settings [6, 18, 14, 15]. It is therefore natural to study these problems through the lens of parameterized complexity [8, 2]. Apart from parameterizing by the solution size (i.e., the number of agents assigned to any activity in a solution) [19], the perhaps most prominent parameterizations thus far have been the number of activities, the number of agents, and in the case of gGasp parameterizations tied to the structure of the network [6, 17, 18, 14, 9]. Consequently, the parameterized complexity of all three variants of Gasp w.r.t. the number of activities and/or the number of agents is now almost completely understood.

Namely, computing a stable assignment for a given instance of Gasp is known to be W[1]-hard and contained in XP parameterized by either the number of activities [6, 17, 15] or the number of agents [18, 15] and known to be fixed-parameter tractable parameterized by both parameters [17, 15]. Even though it has never been explicitly stated, the same results also hold for gGasp when parameterizing by the number of agents as well as when using both parameters. This is because both the XP algorithm for the number of agents as well as the fixed-parameter algorithm for both parameters essentially brute-force over every possible assignment and are hence also able to find a solution for gGasp. Moreover, the fact that gGasp generalizes Gasp implies that the W[1]-hardness result for the number of agents also carries over to gGasp.

On the other hand, if we consider the number of activities as a parameter then gGasp turns out to be harder than Gasp: Gupta et al. ([14]) showed that gGasp is NP-complete even when restricted to instances with a single activity. The hardness of gGasp has inspired a series of tractability results [14, 18, 17] obtained by employing additional restrictions on the structure of the network. One prominent result in this direction has been recently obtained by Gupta et al. ([14]), showing that gGasp is fixed-parameter tractable parameterized by the number of activities if the network has constant treewidth.

Already with the introduction of Gasp [6] the authors argued that instead of putting restrictions on the total number of agents, which can be very large in general, it might be much more useful to consider the number of distinct types of agents. It is easy to imagine a setting with large groups of agents that share the same preferences (for instance due to inherent limitations of how preferences are collected). In contrast to the related parameter number of activity types, where it is known that sGasp remains NP-complete even for a constant number of activity types [6], the complexity of the problems parameterized by the number of agent types (with or without restricting the number of activities) has remained wide open thus far.

Our Results. In this paper we obtain a complete classification of the complexity of GASP and its variants sGASP and gGASP when parameterized by the number of agent types (t) alone, and also when parameterized by t plus the number of activities (a). In particular, for each of the considered problems and parameterizations, we determine whether the problem is in FPT, or W[1]-hard and in XP, or paraNP-hard. One distinguishing feature of our lower- and upper-bound results is that they make heavy use of novel Subset-Sum machinery. Below, we provide a high-level summary of the individual results presented in the paper.

Result 1. sGASP is fixed-parameter tractable when parameterized by $t + a$

This is the only fixed-parameter tractability result presented in the paper, and is essentially tight: it was recently shown that sGASP is W[1]-hard when parameterized by a alone [9], and the W[1]-hardness of the problem when parameterized by t is obtained in this paper. Our first step towards obtaining the desired fixed-parameter algorithm for sGASP is to show that every YES-instance contains a solution which is *acyclic* – in particular, a solution with no cycles formed by interactions between activities and agent types (captured in terms of the *incidence graph* G of an assignment). This is proved via the identification of certain *compression steps* which can be applied on a solution in order to remove cycles.

Once we show that it suffices to focus on acyclic solutions, we branch over all acyclic incidence graphs (i.e., all acyclic edge sets of G); for each such edge set, we can reduce the problem of determining whether there exists an assignment realizing this edge set to a variant of SUBSET SUM embedded in a tree structure. The last missing piece is then to show that this problem, which we call TREE SUBSET SUM, is polynomial-time tractable; this is done via dynamic programming, whereas each step boils down to solving a simplified variant of SUBSET SUM.

Result 2. sGASP is W1-hard when parameterized by t

Our second result complements Result 1. As a crucial intermediate step towards Result 2, we obtain the W[1]-hardness of a variant of SUBSET SUM with three distinct "ingredients":

1. **Partitioning**: items are partitioned into sets, and precisely one item must be selected from each set,
2. **Multidimensionality**: each item is a d-dimensional vector (d being the parameter) where the aim is to hit the target value for each component, and
3. **Simplicity**: each vector contains precisely one non-zero component.

We call this problem SIMPLE MULTIDIMENSIONAL PARTITIONED SUBSET SUM (SMPSS). Note that SMPSS is closely related to MULTIDIMENSIONAL SUBSET SUM (MSS), which (as one would expect) merely enhances SUBSET SUM via Ingredient 2. MSS has recently been used to establish W[1]-hardness for parameterizations of EDGE DISJOINT PATHS [13] and BOUNDED DEGREE VERTEX DELETION [12]. However, Ingredient 1 and especially Ingredient 3 are critical requirements for our reduction to work, and establishing the W[1]-hardness of SMPSS was the main challenge on the way towards the desired lower-bound result for sGASP. Since MSS has already been successfully used to obtain lower-bound results and SMPSS is a much more powerful tool in this regard, we believe that SMPSS will find applications in establishing lower bounds for other problems in the future.

Result 3. Gasp **is in XP when parameterized by** t

This is the only XP result required for our complexity map, as it implies XP algorithms for sGasp parameterized by t and for Gasp parameterized by t. We note that the techniques used to obtain Result 3 are disjoint from those used for Result 1; in particular, our first step is to reduce Gasp parameterized by t to solving "XP-many" instances of sGasp parameterized by t. This is achieved by showing that once we know a "least preferred alternative" for every agent type that is active in an assignment, then the Gasp instance becomes significantly easier – and, in particular, can be reduced to a (slightly modified version of) sGasp. It is interesting to note that the result provides a significant conceptual improvement over the known brute force algorithm for Gasp parameterized by the number of agents which enumerates all possible assignments of agents to activities [16, Theorem 3] (see also [15]): instead of guessing an assignment for all agents, one merely needs to guess a least preferred alternative for every agent type.

The second part of our journey towards Result 3 focuses on obtaining an XP algorithm for sGasp parameterized by t. This algorithm has two components. Initially, we show that in this setting one can reduce sGasp to the problem of finding an assignment which is individually rational (i.e., without the stability condition) and satisfies some additional minor properties. To find such an assignment, we once again make use of Subset Sum: in particular, we develop an XP algorithm for the MPSS problem (i.e., Subset Sum enhanced by ingredients 1 and 2) and apply a final reduction from finding an individually rational assignment to MPSS.

Result 4. Gasp **is W1-hard when parameterized by** $t + a$

Result 5. gGasp **is W1-hard when parameterized by** $t + a$ **and the** ***vertex cover number*** **[11] of the network**

The final two results are hardness reductions which represent the last pieces of the presented complexity map. Both are obtained via reductions from Partitioned Clique (also called Multicolored Clique in the literature [2]), and both reductions essentially use $k + \binom{k}{2}$ activities whose sizes encode the vertices and edges forming a k-clique. The main challenge lies in the design of (a bounded number of) agent types whose preference lists ensure that the chosen vertices are indeed endpoints of the chosen edges. The reduction for gGasp then becomes even more involved, as it can only employ a limited number of connections between the agents in order to ensure that vertex cover of the network is bounded.

We note that Result 5 also followed up on previous work by Gupta, Roy, Saurabh and Zehavi [14], who showed that gGasp is fixed-parameter tractable parameterized by the number of activities if the network has constant treewidth. In this sense, our hardness result represents a substantial shift of the boundaries of (in)tractability: in addition to excluding fixed-parameter tractability when parameterizing by the number of activities and treewidth, it also rules out the use of agent types as a parameter and replaces treewidth by the more restrictive vertex cover number. An overview of our results is provided in Table 1.

2 Preliminaries

For an integer i, we let $[i] = \{1, 2, \ldots, i\}$ and $[i]_0 = [i] \cup \{0\}$. We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{N}_0$ the set $\mathbb{N} \cup \{0\}$. For a set S and an integer k, we denote by S^k and 2^S the set of all k dimensional vectors over S and the set of all subsets of S, respectively.

Table 1 Lower and upper bounds for sGasp, Gasp, and gGasp parameterized by the number of agent types (t), with or without additionally parameterizing by the number of activities (a). In the case of gGasp, also the parameter vertex cover number (vc) of the network is considered. The numbers 1 to 5 in the upper index are used to identify results 1 to 5. Entries in bold are shown in this paper; previously known entries follow from the work of Gupta et al. [14].

	Parameter	Lower Bound	Upper Bound
sGasp		**W[1]**^{2}	**XP**
Gasp	t	**W[1]**	**XP**^{3}
gGasp		paraNP	
sGasp		**FPT**^{1}	
Gasp	$t + a$	**W[1]**^{4}	**XP**
gGasp		paraNP	
gGasp	$t + a +$ vc	**W[1]**^{5}	XP

We refer to the handbook by Diestel ([7]) for standard graph terminology. Due to space constraints, we also refer to the respective handbooks [8, 2] for standard terminology and basic notions in parameterized complexity. The *vertex cover number* of a graph G is the size of a minimum vertex cover of G.

2.1 Group Activity Selection

The task in the Group Activity Selection Problem (Gasp) is to compute a *stable assignment* π from a given set N of *agents* to a set A of *activities*, where each agent participates in at most one activity in A. The assignment π is (Nash) stable if and only if it is *individually rational* and no agent has an *NS-deviation* to any other activity (both of these stability rules are defined in the next paragraph). We use a dummy activity $a_\emptyset$ to capture all those agents that do not participate in any activity in A and denote by A^* the set $A \cup \{a_\emptyset\}$. Thus, an assignment π is a mapping from N to A^*, and for an activity $a \in A$ we use $\pi^{-1}(a)$ to denote the set of agents assigned to a by π; we set $|\pi^{-1}(a_\emptyset)| = 1$ if there is at least one agent assigned to $a_\emptyset$ and 0 otherwise.

The set X of *alternatives* is defined as $X = (A \times [|N|]) \cup \{(a_\emptyset, 1)\}$. Each agent is associated with its own *preferences* defined on the set X. In the case of the standard Gasp problem, an instance I is of the form $(N, A, (\succeq_n)_{n \in N})$ where each agent n is associated with a complete transitive preference relation (list) $\succeq_n$ over the set X. An assignment π is *individually rational* if for every agent $n \in N$ it holds that if $\pi(n) = a$ and $a \neq a_\emptyset$, then $(a, |\pi^{-1}(a)|) \succeq_n (a_\emptyset, 1)$ (i.e., n weakly prefers staying in a over moving to $a_\emptyset$). An agent n where $\pi(n) = a$ is defined to have an *NS-deviation* to a different activity a' in A if $(a', |\pi^{-1}(a')| + 1) \succ_n (a, |\pi^{-1}(a)|)$ (i.e., n prefers moving to an activity a' over staying in a). The task in Gasp is to compute a stable assignment.

gGasp is defined analogously to Gasp, however where one is additionally given a set L of *links* $L \subseteq \{\, \{n, n'\} \mid n, n' \in N \wedge n \neq n' \,\}$ between the agents on the input; specifically, L can be viewed as a set of undirected edges and (N, L) as a simple undirected graph. In gGasp, the task is to find an assignment π which is not only stable but also connected; formally, for every $a \in A$ the set of agents $\pi^{-1}(a)$ induces a connected subgraph of (N, L). Moreover, an agent $n \in N$ only has an NS-deviation to some activity $a \neq \pi(n)$ if (in addition to the conditions for NS-deviations defined above) n has an edge to at least one agent in $\pi^{-1}(a)$.

In sGasp, an instance I is of the form $(N, A, (P_n)_{n\in N})$, where each agent has an *approval set* $P_n \subseteq X \setminus \{(a_\emptyset, 1)\}$ of preferences (instead of an ordered preference list). We denote by $P_n(a)$ the set $\{\, i \mid (a,i) \in P_n \,\}$ for an activity $a \in A$. An assignment $\pi : N \to A^*$ is said to be *individually rational* if every agent $n \in N$ satisfied the following: if $\pi(n) = a$ and $a \neq a_\emptyset$, then $|\pi^{-1}(a)| \in P_n(a)$. Further, an agent $n \in N$ where $\pi(n) = a_\emptyset$, is said to have an *NS-deviation* to an activity a in A if $(|\pi^{-1}(a)| + 1) \in P_n(a)$.

We now introduce the notions and definitions required for our main parameter of interest, the "number of agent types". We say that two agents n and n' in N have the same *agent type* if they have the same preferences. To be specific, $P_n = P_{n'}$ for sGasp and $\succeq_n = \succeq_{n'}$ for Gasp and gGasp. In the case of sGasp and Gasp n and n' are indistinguishable, while in gGasp n and n' can still have different links to other agents. For a subset $N' \subseteq N$, we denote by $T(N')$ the set of agent types occurring in N'. Note that this notation requires that the instance is clear from the context. If this is not the case then we denote by $T(I)$ the set $T(N)$ if N is the set of agents for the instance I of sGasp, Gasp, or gGasp.

For every agent type $t \in T(I)$, we denote by N_t the subset of N containing all agents of type t; observe that $\{\, N_t \mid t \in T(I) \,\}$ forms a partition of N. For an agent type $t \in T(I)$ we denote by P_t (sGasp) or $\succeq_t$ (Gasp) the preference list assigned to all agents of type t and we use $P_t(a)$ (for an activity $a \in A$) to denote P_t restricted to activity a, i.e., $P_t(a)$ is equal to $P_n(a)$ for any agent n of type t. For an assignment $\pi : N \to A^*$, $t \in T(I)$, and $a \in A$ we denote by $\pi_{t,a}$ the set $\{\, n \mid n \in N_t \wedge \pi(n) = a \,\}$ and by π_t the set $\bigcup_{a\in A} \pi_{t,a}$. Further, $\pi(t)$ is the set of all activities that have at least one agent of type t participating in it. We say that π is a perfect assignment for some agent type $t \in T(I)$ if $\pi(n) \neq a_\emptyset$ for every $n \in N_t$. We denote by $\mathrm{PE}(I, \pi)$ the subset of $T(I)$ consisting of all agent types that are perfectly assigned by π, and say that π is a *perfect assignment* if $\mathrm{PE}(I, \pi) = T(I)$.

One notion that will appear through the paper is that of *compatibility*: for a subset $Q \subseteq T(I)$, we say that π is *compatible* with Q if $\mathrm{PE}(I, \pi) = Q$. We conclude this section with a technical lemma which provides a preprocessing procedure that will be used as a basic tool for obtaining our algorithmic results. In particular, Lemma 1 allows us to reduce the problem of computing a stable assignment for a sGasp instance compatible with Q to the problem of finding an individually rational assignment.

▶ **Lemma 1.** *Let $I = (N, A, (P_n)_{n\in N})$ be an instance of* sGasp *and $Q \subseteq T(I)$. Then in time $\mathcal{O}(|N|^2|A|)$ one can compute an instance $\gamma(I, Q) = (N, A, (P'_n)_{n\in N})$ and $A_{\neq\emptyset}(I, Q) \subseteq A$ with the following property: for every assignment $\pi : N \to A^*$ that is compatible with Q, it holds that π is stable for I if and only if π is individually rational for $\gamma(I, Q)$ and $\pi^{-1}(a) \neq \emptyset$ for every $a \in A_{\neq\emptyset}(I, Q)$.*

3 Subset Sum Machinery

In this section we introduce the Subset Sum machinery required for our algorithms and lower bound results. In particular, we introduce three variants of Subset Sum, obtain algorithms for two of them, and provide a W[1]-hardness result for the third.

Tree Subset Sum. Here we introduce a useful generalization of Subset Sum, for which we obtain polynomial-time tractability under the assumption that the input is encoded in unary. Intuitively, our problem asks us to assign values to edges while meeting a simple criterion on the values of edges incident to each vertex.

TREE SUBSET SUM (TSS)

Input:	A vertex-labeled undirected tree T with labeling function $\lambda : V(T) \to 2^{\mathbb{N}}$.
Question:	Is there an assignment $\alpha : E(T) \to \mathbb{N}$ such that for every $v \in V(T)$ it holds that $\sum_{e \in E(T) \wedge v \in e} \alpha(e) \in \lambda(v)$.

Let us briefly comment on the relationship of TSS with SUBSET SUM. Recall that given a set S of natural numbers and a natural number t, the SUBSET SUM problem asks whether there is a subset S' of S such that $\sum_{s \in S'} s = t$. One can easily construct a simple instance (G, λ) of TSS that is equivalent to a given instance (S, t) of SUBSET SUM as follows. G consists of a star having one leaf l_s for every $s \in S$ with $\lambda(l_s) = \{0, s\}$ and $\lambda(c) = \{t\}$ for the center vertex c of the star. Given this simple reduction from SUBSET SUM to TSS it becomes clear that TSS is much more general than SUBSET SUM. In particular, instead of a star TSS allows for the use of an arbitrary tree structure and moreover one can use arbitrary subsets of natural numbers to specify the constrains on the vertices. The above reduction in combination with the fact that SUBSET SUM is weakly NP-hard implies that TSS is also weakly NP-hard. In the remainder of names paragraph we will show that TSS (like SUBSET SUM) can be solved in polynomial-time if the input is given in unary. This will later be used to obtain Result 1 (in Section 4).

Let $I = (T, \lambda)$ be an instance of TSS. We denote by $\max(I)$ the value of the maximum number occurring in any vertex label. The main idea behind our algorithm for TSS is to apply leaf-to-root dynamic programming. In order to execute our dynamic programming procedure, we will need to solve a special case of TSS which we call PARTITIONED SUBSET SUM; this is the problem that will later arise when computing the dynamic programming tables for TSS. In the PARTITIONED SUBSET SUM problem one is given a target set R of natural numbers and ℓ source sets $S_1, \ldots S_\ell$ of natural numbers and the aim is to compute the set S of all natural numbers s such that there are $s_1, \ldots, s_\ell$, where $s_i \in S_i$ for every i with $1 \le i \le \ell$, satisfying $(\sum_{1 \le i \le \ell} s_i) + s \in R$.

▶ **Lemma 2.** *An instance $I = (T, \lambda)$ of* TSS *can be solved in time $\mathcal{O}(|V(T)|^2 \cdot \max(I)^2)$.*

Multidimensional Partitioned Subset Sum. Our second generalization of SUBSET SUM is a multi-dimensional variant of the problem that allows to separate the input set of numbers into several groups, and restricts the solution to take at most 1 vector from each group. For technical reasons, we will only search for solutions of size at most r.

MULTIDIMENSIONAL PARTITIONED SUBSET SUM (MPSS)

Input:	$k \in \mathbb{N}$, $r \in \mathbb{N}_0$, and a family $\mathcal{P} = \{P_1, \ldots P_l\}$ of sets of vectors over $\mathbb{N}_0^k$.
Question:	Compute the set of all vectors $\bar{t} \in \{0, \ldots, r\}^k$ such that there are $\bar{p}_1, \ldots, \bar{p}_l$ with $\bar{p}_i \in P_i$ for every i with $1 \le i \le l$ such that $\sum_{1 \le i \le l} \bar{p}_i = \bar{t}$.

It is easy to see that SUBSET SUM is a special case of MPSS: given an instance of SUBSET SUM, we can create an equivalent instance of MPSS by setting r to a sufficiently large number and simply making each group P_i contain two vectors: the all-zero vector and the vector that is equal to the i-th number of the SUBSET SUM instance in all entries. The following algorithm is used as a subprocedure for Result 3 (in Section 6).

▶ **Lemma 3.** *An instance $I = (k, r, \mathcal{P})$ of* MPSS *can be solved in time $\mathcal{O}(|I| \cdot r^k)$.*

Simple Multidimensional Partitioned Subset Sum. Here, we are interested in a much more restrictive version of MPSS, where all vectors (apart from the target vector) are only allowed to have at most one non-zero component. Surprisingly, we show that the W[1]-hardness of the previously studied Multidimensional Subset Sum problem [13, 12] carries over to this more restrictive variant using an intricate and involved reduction. This result forms the main ingredient needed for our Result 2 (provided in Secction 5). To formalize our problem, we say that a set P of vectors in $\mathbb{N}_0^d$ is *simple* if each vector in P has at most one non-zero component and the values of the non-zero components for any two distinct vectors in P are distinct.

Simple Multidimensional Partitioned Subset Sum (SMPSS)	
Input:	$d \in \mathbb{N}$, $\bar{t} \in \mathbb{N}_0^d$, and a family $\mathcal{P} = \{P_1, \dots P_l\}$ of simple sets of vectors in $\mathbb{N}_0^d$.
Parameter:	d.
Question:	Are there vectors $\bar{p}_1, \dots, \bar{p}_l$ with $\bar{p}_i \in P_i$ for every i with $1 \leq i \leq l$ such that $\sum_{1 \leq i \leq l} \bar{p}_i = \bar{t}$.

▶ **Theorem 4.** SMPSS *is strongly* W[1]*-hard.*

Proof Sketch. We will employ a parameterized reduction from the Partitioned Clique problem, which is well-known to be W[1]-complete [20]. In Partitioned Clique we are given an integer k along with a graph G whose vertex set V is partitioned into k given independent sets $V_1, \dots, V_k$, and are asked to decide whether G contains a k-clique. We denote by $E_{i,j}$ the set of edges of G that have one endpoint in V_i and one endpoint in V_j and we assume w.l.o.g. that $|V_i = \{v_1^i, \dots, v_n^i\}| = n$ and $|E_{i,j}| = m$ for every i and j with $1 \leq i < j \leq k$ (see the standard textbooks for a justification of these assumptions [2, 8]).

Given an instance (G, k) of Partitioned Clique with partition $V_1, \dots, V_k$, we construct an equivalent instance $I = (d, \bar{t}, \mathcal{P})$ of SMPSS in polynomial time, where $d = k(k-1) + \binom{k}{2}$ and $|\mathcal{P}| = \binom{k}{2} + nk(2k-3)$. We will also make use of the following notation. For i and j with $1 \leq i \leq k$ and $1 \leq j < k$, we denote by $\mathsf{indJ}(i, j)$ the j-th smallest number in $[k] \setminus \{i\}$ and we denote by $\mathsf{indMin}(i)$ and $\mathsf{indMax}(i)$ the numbers $\mathsf{indJ}(i, 1)$ and $\mathsf{indJ}(i, k-1)$, respectively.

We assign to every vertex v of G a unique number $\mathcal{S}(v)$ from a Sidon sequence $\mathcal{S}$ of length $|V(G)|$ [10]. A *Sidon sequence* is a sequence of natural numbers such that the sum of each pair of numbers is unique; it can be shown that it is possible to construct such sequences whose maximum value is bounded by a polynomial in its length [1, 10].

To simplify the description of I, we will introduce names and notions to identify both components of vectors and sets in $\mathcal{P}$. Every vector in I has the following components:

- For every i and j with $1 \leq i, j \leq k$ and $i \neq j$, the *vertex component* $c_V^i(j)$. We set $\bar{t}[c_V^i(j)]$ to:
  - $n^6 + n^4$ if $j = \mathsf{indMin}(i)$,
  - $(n-1)n^8 + n^6 + n^4 + \sum_{\ell=1}^{n}(\ell + \ell n^2)$ if $j > \mathsf{indMin}(i)$ and $j < \mathsf{indMax}(i)$, and
  - $(n-1)n^8 + n^6 + \sum_{\ell=1}^{n} \ell$, otherwise.
- For every i and j with $1 \leq i < j \leq k$, the *edge component* $c_E(i,j)$ with $\bar{t}[c_E(i,j)] = \sum_{v \in V_i \cup V_j} \mathcal{S}(v)$.

Note that the total number of components d is equal to $k(k-1) + \binom{k}{2}$ and that for every i with $1 \leq i \leq k$, there are $k-1$ vertex components, i.e., the components $c_V^i(\mathsf{indJ}(i,1))$, $\dots$, $c_V^i(\mathsf{indJ}(i,k-1))$, which intuitively have the following tasks. The first component, i.e., the component $c_V^i(\mathsf{indJ}(i,1))$ identifies a vertex $v \in V_i$ that should be part of a k-clique in G. Moreover, every component $c_V^i(\mathsf{indJ}(i,j))$ (including the first component), is also responsible for: (1) Signalling the choice of the chosen vertex $v \in V_i$ to the next component, i.e., the component $c_V^i(\mathsf{indJ}(i, j + 1))$ and (2) Signalling the choice of the vertex $v \in V_i$ to the component $c_E(i, j)$ that will then verify that there is an edge between the vertex chosen for V_i and the vertex chosen for V_j. This interplay between the components will be achieved through the sets of vectors in $\mathcal{P}$ that will be defined and explained next.

Table 2 An illustration of the vectors contained in the sets $P_{EV}^1(2,\ell), \dots, P_{EV}^1(4,\ell)$ and $P_V^1(2,\ell), \dots, P_V^1(3,\ell)$. For example the column for the set $P_{EV}^1(2,\ell)$ shows that the set contains two vectors, one whose only non-zero component $c_V^1(2)$ has the value $n^6 + \ell$ and a second one whose only non-zero component $c_E(1,2)$ and has the value $\mathcal{S}(v_\ell^1)$. The last column provides the value for the target vector for the component given by the row. Finally, the value Z is equal to $(n-1)n^8 + n^6 + \sum_{l=1}^{n}(\ell)$.

	$P_{EV}^1(2,\ell)$	$P_V^1(2,\ell)$	$P_{EV}^1(3,\ell)$	$P_V^1(3,\ell)$	$P_{EV}^1(4,\ell)$	$\bar{t}$
$c_V^1(2)$	$n^6+\ell$	$n^4-\ell$				n^6+n^4
$c_V^1(3)$		$n^8+\ell+\ell n^2$	$n^6+\ell$	$n^4+\ell n^2$		$Z+n^4+\sum_{\ell=1}^{n}(\ell n^2)$
$c_V^1(4)$				$n^8+\ell$	$n^6+\ell$	Z
$c_E(1,2)$	$\mathcal{S}(v_\ell^1)$					$\sum_{v\in V_1\cup V_2}\mathcal{S}(v)$
$c_E(1,3)$			$\mathcal{S}(v_\ell^1)$			$\sum_{v\in V_1\cup V_3}\mathcal{S}(v)$
$c_E(1,4)$					$\mathcal{S}(v_\ell^1)$	$\sum_{v\in V_1\cup V_4}\mathcal{S}(v)$

Table 3 An illustration of the vectors contained in the sets $P_{EV}^i(j,\ell)$, $P_{EV}^j(i,\ell)$, and $P_E(i,j)$ and their interplay with the components $c_V^i(j)$, $c_V^j(i)$, and $c_V(i,j)$. For the conventions used in the table please refer to Table 2. Additionally, note that the column for $P_E(i,j)$ indicates that the set contains one vector for every edge $\{v,u\} \in E_{i,j}$, whose only non-zero component $c_E(i,j)$ has the value $\mathcal{S}(v) + \mathcal{S}(u)$.

	$P_{EV}^i(j,\ell)$	$P_{EV}^j(i,\ell)$	$P_E(i,j)$	$\bar{t}$
$c_V^i(j)$	$n^6+\ell$			
$c_V^j(i)$		$n^6+\ell$		
$c_E(i,j)$	$\mathcal{S}(v_\ell^i)$	$\mathcal{S}(v_\ell^j)$	$\{\mathcal{S}(v)+\mathcal{S}(u) \mid \{v,u\}\in E_{i,j}\}$	$\sum_{v\in V_i\cup V_j}\mathcal{S}(v)$

$\mathcal{P}$ consists of the following sets, which are illustrated in Table 2 and 3:

- For every i, j', and ℓ with $1 \le i \le k$, $1 \le j' \le k-2$, and $1 \le \ell \le n$, the *vertex set* $P_V^i(j,\ell)$, where $j = \mathsf{indJ}(i,j')$, containing two vectors $\bar{v}_{i,j,\ell}^+$ and $\bar{v}_{i,j,\ell}^-$ defined as follows:
  - if $j' = 1$, then $\bar{v}_{i,j,\ell}^+[c_V^i(j)] = n^4 - \ell$ and $\bar{v}_{i,j,\ell}^-[c_V^i(\mathsf{indJ}(i,j'+1))] = n^8 + \ell + \ell n^2$ or
  - if $1 < j' < k-2$, then $\bar{v}_{i,j,\ell}^+[c_V^i(j)] = n^4 + \ell n^2$ and $\bar{v}_{i,j,\ell}^-[c_V^i(\mathsf{indJ}(i,j'+1))] = n^8+\ell+\ell n^2$ or
  - if $j' = k-2$, then $\bar{v}_{i,j,\ell}^+[c_V^i(j)] = n^4 + \ell n^2$ and $\bar{v}_{i,j,\ell}^-[c_V^i(\mathsf{indJ}(i,j'+1))] = n^8 + \ell$.

  We denote by $P_V^i(j)$, $P_{V+}^i(j)$, and $P_{V-}^i(j)$ the sets $\bigcup_{\ell=1}^n(P_V^i(j,\ell))$, $P_V^i(j) \cap \{\, \bar{v}_{i,j,\ell}^+ \mid 1 \le \ell \le n \,\}$, and $P_V^i(j) \setminus P_{V+}^i(j)$, respectively.
- For every i, j, and ℓ with $1 \le i,j \le k$, $i \ne j$, and $1 \le \ell \le n$, the *vertex incidence set* $P_{EV}^i(j,\ell)$, which contains the two vectors $\bar{a}_{i,j,\ell}^+$ and $\bar{a}_{i,j,\ell}^-$ such that $\bar{a}_{i,j,\ell}^+[c_V^i(j)] = n^6 + \ell$ and $\bar{a}_{i,j,\ell}^-[c_E(i,j)] = \mathcal{S}(v_\ell^i)$. We denote by $P_{EV}^i(j)$, $P_{EV+}^i(j)$, and $P_{EV-}^i(j)$ the sets $\bigcup_{\ell=1}^n(P_{EV}^i(j,\ell))$, $P_V^i(j) \cap \{\, \bar{a}_{i,j,\ell}^+ \mid 1 \le \ell \le n \,\}$, and $P_{EV}^i(j) \setminus P_{EV+}^i(j)$, respectively.
- For every i, j with $1 \le i < j \le k$, the *edge set* $P_E(i,j)$, which for every $e = \{v,u\} \in E_{i,j}$ contains the vector $\bar{e}$ such that $\bar{e}[c_E(i,j)] = \mathcal{S}(v) + \mathcal{S}(u)$; note that $P_E(i,j)$ is indeed a simple set, because $\mathcal{S}$ is a Sidon sequence.

Note that altogether there are $nk(k-2) + \binom{k}{2} + nk(k-1) = \binom{k}{2} + nk(2k-3)$ sets in $\mathcal{P}$.

Informally, the two vectors $\bar{v}^+_{i,j,\ell}$ and $\bar{v}^-_{i,j,\ell}$ in $P^i_V(j,\ell)$ represent the choice of whether or not the vertex v^i_ℓ should be included in a k-clique for G, i.e., if a solution for I chooses $v^+_{i,j,\ell}$ then v^i_ℓ should be part of a k-clique and otherwise not. The component $c^i_V(j)$, more specifically the value for $\bar{t}[c^i_V(j)]$, now ensures that a solution can choose at most one such vector in $P^i_{V+}(j)$. Moreover, the fact that all but one of the vectors $\bar{v}^-_{i,j,1},\ldots,\bar{v}^-_{i,j,n}$ need to be chosen by a solution for I signals the choice of the vertex for V_i to the next component, i.e., either the component $c^i_V(j+1)$ if $j+1\neq i$ or the component $c^i_V(j+2)$ if $j+1=i$. Note that we only need $k-2$ sets $P^i_V(j)$ for every i, because we need to copy the vertex choice for V_i to only $k-1$ components. A similar idea underlies the two vectors $\bar{a}^+_{i,j,\ell}$ and $\bar{a}^-_{i,j,\ell}$ in $P^i_{EV}(j,\ell)$, i.e., again the component $c^i_V(j)$ ensures that $\bar{a}^+_{i,j,\ell}$ can be chosen for only one of the sets $P^i_{EV}(j,1),\ldots,P^i_{EV}(j,n)$ and $\bar{a}^-_{i,j,\ell}$ must be chosen for all the remaining ones. Note that the component $c^i_V(j)$ now also ensures that the choice made for the sets in $P^i_V(j)$ is the same as the choice made for the sets in $P^i_{EV}(j)$. Moreover, the choice made for the sets in $P^i_{EV}(j)$ is now propagated to the component $c_E(i,j)$ (instead of the next vertex component). Finally, the vectors in the set $P_E(i,j)$ represent the choice of the edge used in a k-clique between V_i and V_j and the component $c_E(i,j)$ ensures that only an edge, whose endpoints are the two vertices signalled by the sets $P^i_{EV}(j)$ and $P^j_{EV}(i)$ can be chosen. ◀

4 Result 1: Fixed-Parameter Tractability of sGasp

In this section we will establish that sGasp is FPT when parameterized by the number of agent types and the number of activities by proving Theorem 5.

▶ **Theorem 5.** sGasp *can be solved in time* $\mathcal{O}(2^{|T(N)|\cdot(1+|A|)}\cdot((|N|+|A|)|N|)^2)$.

Let $I=(N,A,(P_n)_{n\in N})$ be a sGasp instance and let $\pi : N\to A^*$ be an assignment of agents to activities. We denote by $G_I(\pi)$ the incidence graph between $T(N)$ and A, which is defined as follows. $G_I(\pi)$ has vertices $T(N)\cup A$ and contains an edge between an agent type $t\in T(N)$ and an activity $a\in A$ if $\pi_{t,a}\neq\emptyset$. We say that π is *acyclic* if $G_I(\pi)$ is acyclic.

Our first aim towards the proof of Theorem 5 is to show that if I has a stable assignment, then it also has an acyclic stable assignment (Lemma 7). We will then show in Lemma 9 that finding a stable assignment whose incidence graph is equal to some given acyclic pattern graph can be achieved in polynomial-time via a reduction to the TSS problem (see Lemma 2). Since the number of (acyclic) pattern graphs is bounded in our parameters, we can subsequently solve sGasp by enumerating all acyclic pattern graphs and checking for each of them whether there is an acyclic solution matching the selected pattern.

A crucial notion towards showing that it is sufficient to consider only acyclic solutions is the notion of (strict) compression. We say that an assignment τ is a *compression* of π if it satisfies the following conditions:

(C1) for every $t\in T(N)$ it holds that $|\pi_t|=|\tau_t|$,

(C2) for every $a\in A$ it holds that $|\pi^{-1}(a)|=|\tau^{-1}(a)|$, and

(C3) for every $a\in A$ it holds that the set of agent types τ assigns to a is a subset of the agent types π assigns to a.

Intuitively, an assignment τ is a compression of π if it maintains all the properties required to preserve stability and compatibility with a given subset $Q\subseteq T(N)$. We note that condition (C3) can be formalized as $T(\pi^{-1}(a))\subseteq T(\pi^{-1}(a))$. The following lemma shows that every assignment that is not acyclic admits a compression.

▶ **Lemma 6.** *Let* $\pi : N\to A^*$ *be an assignment for* I*. Then there exists an acyclic assignment* π' *that compresses* π*.*

The next lemma provides the first cornerstone for our algorithm by showing that it is sufficient to consider only acyclic solutions. Intuitively, it is a consequence of Lemma 6 along with the observation that compression preserves stability and individual rationality.

▶ **Lemma 7.** *If I has a stable assignment, then I has an acyclic stable assignment.*

Our next step is the introduction of terminology related to the pattern graphs mentioned at the beginning of this section. Let G be a bipartite graph with bi-partition $\{T(N), A\}$. We say that G *models* an assignment $\pi : N \to A^*$ if $G_I(\pi) = G$; in this sense every such bipartite graph can be seen as a pattern (or model) for assignments. For a subset $Q \subseteq T(N)$ we say that G is *compatible* with Q if every vertex in Q and every vertex in $A_{\neq\emptyset}(I, Q)$ (recall the definition of $A_{\neq\emptyset}(I, Q)$ given in Lemma 1) has at least one neighbor in G; note that if G is compatible with Q then any assignment π modeled by G satisfies $\tau^{-1}(a) \neq \emptyset$ for every $a \in A_{\neq\emptyset}(I, Q)$. Intuitively, the graph G captures information about which types of agents are mapped to which activities (without specifying numbers), while Q captures information about which agent types are perfectly (i.e., "completely") assigned.

Let $Q \subseteq T(N)$ and let G be a bipartite graph with bi-partition $\{T(N), A\}$ that is compatible with Q. The following simple lemma shows that, modulo compatibility requirements, finding a stable assignment for I can be reduced to finding an individually rational assignment for $\gamma(I, Q)$ (recall the definition of $\gamma(I, Q)$ given in Lemma 1).

▶ **Lemma 8.** *Let $Q \subseteq T(N)$ and let G be a bipartite graph with bi-partition $\{T(N), A\}$ that is compatible with Q. Then for every assignment $\pi : N \to A^*$ modeled by G and compatible with Q, π is stable for I if and only if π is individually rational for $\gamma(I, Q)$.*

The next lemma forms (along with Lemma 7) the core component for our proof.

▶ **Lemma 9.** *Let $Q \subseteq T(N)$ and let G be an acyclic bipartite graph with bi-partition $\{T(N), A\}$ that is compatible with Q. Then one can decide in time $\mathcal{O}((|N| + |A|)^2|N|^2)$ whether I has a stable assignment which is modeled by G and compatible with Q.*

We now have all the ingredients needed to establish Theorem 5 ($\star$).

5 Result 2: Lower Bound for sGasp

In this section we complement Theorem 5 by showing that if we drop the number of activities in the parameterization, then sGasp becomes W[1]-hard. We achieve this via a parameterized reduction from SMPSS that we have shown to be strongly W[1]-hard in Theorem 4.

▶ **Theorem 10.** sGasp *is* W[1]*-hard parameterized by the number of agent types.*

6 Result 3: XP Algorithms for sGasp and Gasp

In this section, we present our XP algorithm for Gasp parameterized by the number of agent types. In order to obtain this result, we observe that the stability of an assignment for Gasp can be decided by only considering the stability of agents that are assigned to a "minimal alternative" w.r.t. their type. We then show that once one guesses (i.e., branches over) a minimal alternative for every agent type, the problem of finding a stable assignment for Gasp that is compatible with this guess can be reduced to the problem of finding a perfect and individual rational assignment for a certain instance of sGasp, where one additionally requires that certain activities are assigned to at least one agent. Our first task will hence be to obtain an XP algorithm which can find such a perfect and individually rational assignment for sGasp. To that end, we obtain Lemma 11, which allows us to find certain individually rational assignments in sGasp instances and forms a core part of our XP algorithm for Gasp.

▶ **Lemma 11.** *Let $I = (N, A, (P_n)_{n\in N})$ be an instance of* sGasp*, $Q \subseteq T(N)$, and $A_{\neq\emptyset} \subseteq A$. Then one can decide in time $\mathcal{O}(|A| \cdot (|N|)^{|T(N)|})$ whether I has an individual rational assignment π that is compatible with Q such that $\pi^{-1}(a) \neq \emptyset$ for every $a \in A_{\neq\emptyset}$.*

As a secondary result, we can already obtain an XP algorithm for sGasp parameterized by the number of agent types, which may also be of independent interest, as the obtained running time is strictly better than that of the algorithm obtained for the more general Gasp.

▶ **Theorem 12.** *An instance $I = (N, A, (P_n)_{n\in N})$ of* sGasp *can be solved in time $|A| \cdot |N|^{\mathcal{O}(|T(N)|)}$.*

Our next aim is to use Lemma 11 to obtain an XP algorithm for Gasp. To simplify the presentation of our algorithm, we start by introducing the notion of an NS*-deviations that combines and unifies individual rationality and NS-deviations. Namely, let $I = (N, A, (\succeq_n)_{n\in N})$ be a Gasp instance, $\pi : N \to A^*$ be an assignment, and $n \in N$. We say that n has an NS*-deviation to an activity $a' \in A^* \setminus \{\pi(n)\}$ if $(a', |\pi^{-1}(a')| + 1) \succ_{T(n)} (a, |\pi^{-1}(a)|)$. In order to deal with the case that $a' = a_\emptyset$, we let $(a_\emptyset, i + 1)$ stand for $(a_\emptyset, 1)$ for every i.

▶ **Observation 13.** An assignment π for I is stable if and only if no agent $n \in N$ has an NS*-deviation to any activity in $A^* \setminus \{\pi(n)\}$.

Let I and π be as above and let $t \in T(I)$. We denote by π^*_t the set of activities π_t if t is perfectly assigned by π and $\pi_t \cup \{a_\emptyset\}$, otherwise. We say an activity $a \in \pi^*_t$ is *minimal with respect to t* if $(a', |\pi^{-1}(a')|) \succeq_t (a, |\pi^{-1}(a)|)$ for each $a' \in \pi^*_t$ and we address the alternative $(a, |\pi^{-1}(a)|)$ as a *minimal alternative* with respect to t.

The following lemma uses Observation 13 and allows us to characterize the stability condition of an assignment in terms of minimial activities for each agent type.

▶ **Lemma 14.** *An assignment π for I is stable if and only if for each $t \in T(N)$ and each $a \in A^* \setminus \{a_m\}$, it holds that $(a_m, |\pi^{-1}(a_m)|) \succeq_t (a, |\pi^{-1}(a)| + 1)$, where a_m is a minimal activity w.r.t. t.*

The next theorem now employs the above lemma to construct an instance I' of sGasp together with a subset $A_{\neq\emptyset}$ of activities such that for every function $f_{\min} : T(I) \to X$ (or in other words for every guess of minimal alternatives in an assignment), it holds that I has a stable assignment such that $f_{\min}(t)$ is a minimal alternative w.r.t. t for every $t \in T(I)$ if and only if I' has a perfect and individual rational assignment π such that $\pi^{-1}(a) \neq \emptyset$ for every $a \in A_{\neq\emptyset}$. For brevity, we will say that an assignment π is *compatible with* $f_{\min}$ if and only if $f_{\min}(t)$ is a minimal alternative w.r.t. t for every $t \in T(I)$.

▶ **Theorem 15.** *Let $I = (N, A, (\succeq_n)_{n\in N})$ be an instance of* Gasp *and let $f_{\min} : T(N) \to X$, which informally represents a guess of a minimal alternative for every agent type. Then one can in time $\mathcal{O}(|N|^2|A|)$ construct an instance $I' = (N, A \cup \{a_\phi\}, (P_n)_{n\in N})$ of* sGasp *together with a subset $A_{\neq\emptyset}$ of activities such that $|T(I')| \leq 2|T(I)|$ and I has a stable assignment compatible with $f_{\min}(t)$ if and only if I' has a perfect individual rational assignment π with $\pi^{-1}(a) \neq \emptyset$ for every $a \in A_{\neq\emptyset}$.*

We now have all the ingredients needed to prove the main result of this section.

▶ **Theorem 16.** *An instance $I = (N, A, (\succeq_n)_{n\in N})$ of* Gasp *can be solved in time $(|A| \cdot |N|)^{\mathcal{O}(|T(I)|)}$.*

Proof Sketch. Given an instance $I = (N, A, (\succeq_n)_{n \in N})$ of Gasp, the algorithm enumerates all of the at most $(|A| \cdot |N|)^{|T(I)|}$ possible functions $f_{\min}$ and for each such function $f_{\min}$ the algorithm uses Theorem 15 to construct the instance $I' = (N, A \cup \{a_\phi\}, (P_n)_{n \in N})$ of sGasp with $|T(I')| \leq |T(I)|$ together with the set $A_{\neq\emptyset}$ of activities in time $\mathcal{O}(|N|^2|A|)$. It then uses Lemma 11 to decide whether I' has a perfect individual rational assignment π_1 such that $\pi_1^{-1}(a) \neq \emptyset$ for every $a \in A_{\neq\emptyset}$ in time $\mathcal{O}((|A|+1)(|N|)^{|T(I')|}) = \mathcal{O}((|A|+1)(|N|^{2|T(I)|})$. If this is true for at least one of the functions $f_{\min}$, the algorithm returns that I has a solution, otherwise the algorithm correctly returns that I has no solution. ◀

7 Results 4 and 5: Two Lower Bounds

Our next result shows that Gasp is unlikely to be fixed-parameter tractable parameterized by both the number of activities (a) and the number of agent types (t).

▶ **Theorem 17.** Gasp *is* W[1]*-hard parameterized by* $t + a$*.*

Since Gasp and gGasp are equivalent on complete networks the above hardness result clearly also applies to gGasp. However, to our surprise, the hardness does even hold if we additionally parameterize gGasp with the vertex cover number (vc) of the network.

▶ **Theorem 18.** *g*Gasp *is* W[1]*-hard parameterized by* $t + a +$ vc*.*

8 Conclusion

We obtained a comprehensive picture of the parameterized complexity of Group Activity Selection problems parameterized by the number of agent types, both with and without the number of activities as an additional parameter. Our positive results suggest that using the number of agent types is a highly appealing parameter for Gasp and its variants; indeed, this parameter will often be much smaller than the number of agents due to the way preference lists are collected or estimated (as also argued in initial work on Gasp [6]). For instance, in the large-scale event management setting of Gasp (or sGasp), one would expect that preference lists for event participants are collected via simple questionnaires – and so the number of agent types would remain fairly small regardless of the size of the event.

We believe that the techniques used to obtain the presented results, and especially the Subset Sum tools developed to this end, are of broad interest to the algorithms community. For instance, Multidimensional Subset Sum (MSS) has been used as a starting point for W[1]-hardness reductions in at least two different settings over the past year [13, 12], but the simple and partitioned variant of the problem (i.e., SMPSS) is much more restrictive and hence forms a strictly better starting point for any such reductions in the future. This is also reflected in our proof of the W[1]-hardness of SMPSS, which is *significantly* more involved than the analogous result for MSS. Likewise, we expect that the developed algorithms for Tree Subset Sum and Multidimensional Partitioned Subset Sum may find applications as subroutines for (parameterized and/or classical) algorithms in various settings.

Note that there is now an almost complete picture of the complexity of Group Activity Selection problems w.r.t. any combination of the parameters number of agents, number of activities, and number of agent types (see also Table 1). There is only one piece missing, namely, the parameterized complexity of sGasp parameterized by the number of agents, which we resolve for completeness with the following theorem.

▶ **Theorem 19.** sGasp *is fixed-parameter tractable parameterized by the number of agents.*

Proof. Let $I = (N, A, (P_n)_{n \in N})$ be a sGasp instance. The main idea behind the algorithm is to guess (i.e., branch over) the set $M_\emptyset$ of agents that are assigned to $a_\emptyset$ as well as a partition $\mathcal{M}$ of the remaining agents, i.e., the agents in $N \setminus M_\emptyset$, and then check whether there is a stable assignment π for I such that:

(P1) $\pi^{-1}(a_\emptyset) = M_\emptyset$ and

(P2) $\{\, \pi^{-1}(a) \mid a \in A \,\} \setminus \{\emptyset\} = \mathcal{M}$, i.e., $\mathcal{M}$ corresponds to the grouping of agents into activities by π.

Since there are at most n^n possibilities for $M_\emptyset$ and $\mathcal{M}$ and those can be enumerated in time $\mathcal{O}(n^n)$, it remains to show how to decide whether there is a stable assignent for I satisfying (P1) and (P2) for any given $M_\emptyset$ and $\mathcal{M}$. Towards showing this, we first consider the implications for a stable assignment resulting from assigning the agents in $M_\emptyset$ to $a_\emptyset$. Namely, let P'_n for every $n \in N$ be the approval set obtained from P_n after removing all alternatives (a, i) such that $i \neq 0$ and there is an agent $n_\emptyset \in M_\emptyset$ with $(a, i+1) \in P_{n_\emptyset}$. Moreover, let $A_{\neq\emptyset}$ be the set of all activities that cannot be left empty if the agents in $M_\emptyset$ are assigned to $a_\emptyset$, i.e., the set of all activities such that there is an agent $n_\emptyset \in M_\emptyset$ with $(a, 1) \in P_{n_\emptyset}$. Now consider a set $M \in \mathcal{M}$, and observe that the set A_M of activities that the agents in M can be assigned to in any stable assignment satisfying (P1) and (P2) is given by: $A_M = \{\, a \mid |M| \in \bigcap_{n \in M} P_n(a)' \,\}$. Let B be the bipartite graph having $\mathcal{M}$ on one side and A on the other side and having an edge between a vertex $M \in \mathcal{M}$ and a vertex $a \in A$ if $a \in A_M$. We claim that I has a stable assignment satisfying (P1) and (P2) if and only if B has a matching that saturates $\mathcal{M} \cup A_{\neq\emptyset}$. Since deciding the existence of such a matching can be achieved in time $\mathcal{O}(\sqrt{|V(B)|}|E(B)|) = \mathcal{O}(\sqrt{|N \cup A|}|N||A|)$ (see e.g. [13, Lemma 4]), establishing this claim is the last component required for the proof of the theorem.

Towards showing the forward direction, let π be a stable assignment for I satisfying (P1) and (P2). Then $O = \{\, \{a, \phi^{-1}(a)\} \mid a \in A \,\}$ is a matching in B that saturates $\mathcal{M} \cup A_{\neq\emptyset}$. Note that O saturates $\mathcal{M}$ due to (P2), moreover, O saturates $A_{\neq\emptyset}$ since otherwise there would be an activity $a \in A_{\neq\emptyset}$ with $\pi^{-1}(a) = \emptyset$, which due to the definition of $A_{\neq\emptyset}$ and (P1) implies there is an agent n with $\pi(n) = a_\emptyset$ such that $1 \in P_n(a)$, contradicting our assumption that π is stable.

Towards showing the reverse direction, let O be a matching in B that saturates $\mathcal{M} \cup A$. Then the assignment π mapping all agents in M (for every $M \in \mathcal{M}$) to its partner in O and all other agents to $a_\emptyset$ clearly already satisfies (P1) and (P2). It remains to show that it is also stable. Note that π is individually rational because of the construction of B. Moreover, assume for a contradiction that there is an agent $n \in N_\emptyset$ with $\pi(n) = a_\emptyset$ and an activity $a \in A$ such that $(a, |\pi^{-1}(a)| + 1) \in P_n$. If $|\pi^{-1}(a)| = 0$, then $a \in A_{\neq\emptyset}$ and hence $|\pi^{-1}(a)| > 0$ (because O saturates $A_{\neq\emptyset}$), a contradiction. If on the other hand $|\pi^{-1}(a)| \neq 0$, then $\{M, a\} \in O$ (for some $M \in \mathcal{M}$), but $(a, |\pi^{-1}(a)|) \notin P'_n$ and hence $\{M, a\} \notin E(B)$, also a contradiction. ◀

For future work, we believe that it would be interesting to see how the complexity map changes if one were to consider the number of activity types instead of the number of activities in our parameterizations.

References

1 Martin Aigner and Günter M. Ziegler. *Proofs from the Book (3. ed.).* Springer, 2004.

2 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

3 Andreas Darmann. Group Activity Selection from Ordinal Preferences. In Toby Walsh, editor, *Algorithmic Decision Theory - 4th International Conference, ADT 2015, Lexington, KY, USA, September 27-30, 2015, Proceedings*, volume 9346 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2015.

4 Andreas Darmann, Janosch Döcker, Britta Dorn, Jérôme Lang, and Sebastian Schneckenburger. On Simplified Group Activity Selection. In Jörg Rothe, editor, *Algorithmic Decision Theory - 5th International Conference, ADT 2017, Luxembourg, Luxembourg, October 25-27, 2017, Proceedings*, volume 10576 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2017.

5 Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard Woeginger. Group activity selection problem with approval preferences. *International J. of Game Theory*, pages 1–30, 2017.

6 Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard J. Woeginger. Group Activity Selection Problem. In *Internet and Network Economics - 8th International Workshop, WINE 2012*, volume 7695 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2012.

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer Verlag, 2013. `doi:10.1007/978-1-4471-5559-1`.

9 Eduard Eiben, Robert Ganian, and Sebastian Ordyniak. A Structural Approach to Activity Selection. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 203–209. ijcai.org, 2018.

10 Paul Erdős and Paul Turán. On a problem of Sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society*, 1(4):212–215, 1941.

11 Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph Layout Problems Parameterized by Vertex Cover. In *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, pages 294–305, 2008.

12 Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On Structural Parameterizations of the Bounded-Degree Vertex Deletion Problem. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, pages 33:1–33:14, 2018.

13 Robert Ganian, Sebastian Ordyniak, and Ramanujan Sridharan. On Structural Parameterizations of the Edge Disjoint Paths Problem. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand*, volume 92 of *LIPIcs*, pages 36:1–36:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

14 Sushmita Gupta, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. Group Activity Selection on Graphs: Parameterized Analysis. In Vittorio Bilò and Michele Flammini, editors, *Algorithmic Game Theory - 10th International Symposium, SAGT 2017, L'Aquila, Italy, September 12-14, 2017, Proceedings*, volume 10504 of *Lecture Notes in Computer Science*, pages 106–118. Springer, 2017.

15 Ayumi Igarashi, Robert Bredereck, and Edith Elkind. On Parameterized Complexity of Group Activity Selection Problems on Social Networks. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS 2017, pages 1575–1577. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

16 Ayumi Igarashi, Robert Bredereck, and Edith Elkind. On Parameterized Complexity of Group Activity Selection Problems on Social Networks. *CoRR*, abs/1703.01121, 2017. `arXiv:1703.01121`.

17 Ayumi Igarashi, Robert Bredereck, Dominik Peters, and Edith Elkind. Group Activity Selection on Social Networks. *CoRR*, abs/1712.02712, 2017. `arXiv:1712.02712`.

18 Ayumi Igarashi, Dominik Peters, and Edith Elkind. Group Activity Selection on Social Networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 565–571. AAAI Press, 2017.

19 Hooyeon Lee and Virginia Vassilevska Williams. Parameterized Complexity of Group Activity Selection. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017*, pages 353–361. ACM, 2017.

20 Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. of Computer and System Sciences*, 67(4):757–771, 2003.