Solving Woeginger's Hiking Problem: Wonderful Partitions in Anonymous Hedonic Games

Andrei Constantinescu[1]
ETH Zürich, Zürich, Switzerland

Pascal Lenzner
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

Rebecca Reiffenhäuser
University of Amsterdam, Amsterdam, The Netherlands

Daniel Schmand
University of Bremen, Bremen, Germany

Giovanna Varricchio
University of Calabria, Rende, Italy

Abstract

A decade ago, Gerhard Woeginger posed an open problem that became well-known as "Woeginger's Hiking Problem": Consider a group of n people that want to go hiking; everyone expresses preferences over the size of their hiking group in the form of an interval between 1 and n. Is it possible to efficiently assign the n people to a set of hiking subgroups so that every person approves the size of their assigned subgroup? The problem is also known as efficiently deciding if an instance of an anonymous Hedonic Game with interval approval preferences admits a wonderful partition.

We resolve the open problem in the affirmative by presenting an $O(n^5)$ time algorithm for Woeginger's Hiking Problem. Our solution is based on employing a dynamic programming approach for a specific rectangle stabbing problem from computational geometry. Moreover, we propose natural, more demanding extensions of the problem, e.g., maximizing the number of satisfied participants and variants with single-peaked preferences, and show that they are also efficiently solvable. Last but not least, we employ our solution to efficiently compute a partition that maximizes the egalitarian welfare for anonymous single-peaked Hedonic Games.

2012 ACM Subject Classification Theory of computation → Dynamic programming; Theory of computation → Algorithmic game theory; Theory of computation → Discrete optimization

Keywords and phrases Algorithmic Game Theory, Dynamic Programming, Anonymous Hedonic Games, Single-Peaked Preferences, Social Optimum, Wonderful Partitions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.42

Category Track A: Algorithms, Complexity and Games

Funding *Giovanna Varricchio*: The author is grateful for the support by the PNRR MUR project FAIR - Future AI Research (PE00000013), and the DFG, German Research Foundation, grant (Ho 3831/5-1).

Acknowledgements We would like to thank the Schloss Dagstuhl-Leibniz-Zentrum für Informatik who hosted the event "Computational Social Dynamics" (Seminar 22452) in November 2022 [25], where the work leading to this paper was started. We are grateful to Martin Hoefer, Sigal Oren, and Roger Wattenhofer for organizing this event. We additionally thank Roger Wattenhofer for the useful discussions concerning this work. We thank Tamio-Vesa Nakajima for contributing to the hardness proof in Appendix B.

[1] Corresponding author.

Figure 1 Gerhard Woeginger, Oberwolfach, 2011. [29]

1 Introduction

Suppose there are n attendees of a workshop, who aim to go for a joint hike during a break. Keeping the whole group together is logistically challenging, so typically the attendees split into smaller subgroups that will do the hike together. It is natural that different attendees might have different preferences for the sizes of the subgroups they will eventually join. In particular, each attendee i reports an interval $[\ell_i, r_i]$ signifying that they will be content if they are in a group of size $s \in [\ell_i, r_i]$. The organizers of the hike now face the following problem: Is there a polynomial time algorithm that determines whether there is a partition of the attendees into subgroups such that they are all content with the sizes of their subgroups?

In 2013, Gerhard Woeginger famously phrased this problem underlying an anonymous hedonic game to explain an open question stemming from his work [31]. It is one of the very nice problems that he used to share at various occurrences, e.g., at the coffee machine, while waiting in a seminar room, or even during a joint walk.[2] As this is one of the problems he explained to many people, it became known as Woeginger's *Hiking Problem.*

In this work we answer Woeginger's question in the affirmative by exploiting a tight connection to a variant of the rectangle stabbing problem from computational geometry that can be solved in polynomial time via elegant dynamic programming.

Using Woeginger's motivation of the problem, many natural extensions arise that we introduce in this paper. If the sought partition does not exist, then what is the maximum number of attendees that can be satisfied with their group sizes, or what is the minimum number of attendees to exclude such that the remaining ones can be partitioned to all become satisfied? Moreover, we also consider a version where the hikers have single-peaked preferences over the group sizes and a partition is sought that minimizes the utilitarian or egalitarian cost, where cost is defined as a function of the assigned and the ideal group size. We show that these more demanding problems can also be solved efficiently. Finally, we discuss the relationship between the hiking problem and the problem of maximizing the egalitarian welfare in (general) anonymous Hedonic Games.

[2] Gerhard's ability to explain open research questions in an easily accessible way has been extraordinarily motivating. In particular, this work would not have started without Gerhard meeting one of the authors for lunch and explaining the problem exactly in this way. With this work we contribute to the recent line of publications celebrating the life and work of Gerhard Woeginger, see [26].

1.1 Related Work

Hedonic Games (HGs), introduced by Dreze and Greenberg in [18], model multi-agent systems where selfish agents have to be partitioned into coalitions and have preferences over the possible outcomes. Such games are called hedonic as agents' preferences only depend on the coalition they belong to but not on how the other agents are grouped. HGs have been widely studied (see [7] for a survey) and numerous prominent subclasses have been identified based on properties of the agents' preferences or other possible constraints: [8, 14, 1, 27, 20, 2]. Simple examples of such classes are anonymous HGs [9], where the preferences of the agents depend only on the size of their coalition and not on the individual participants, or HGs with approval-based (Boolean) preferences [6], where agents have binary values for the coalitions. Woeginger's Hiking Problem resides in the intersection of these classes.

The HGs literature has typically focused on the existence and computation of stable or optimal solutions, see, e.g., [14]. The most desirable, ideal partition is the one where every agent is assigned to one of her best coalitions — called *perfect* (or *wonderful*) *partition* [3]. Unfortunately, such a solution rarely exists and the related decision or computational problem is usually hard [3, 28]. Even in simple cases such as anonymous approval-based HGs, it is NP-complete to determine the existence of a wonderful partition [17, 31]. Such a result holds true even if the number of approved coalition sizes of each agent is at most 2 [17]. The problem of finding a wonderful partition in Hedonic Games is related to the one of maximizing the utilitarian welfare, i.e. the sum of agents' utilities, for Boolean utilities. An overall picture of the complexity of finding wonderful partitions in Boolean Hedonic Games, including anonymous ones, is given in [28]. For general utility functions, the problem of maximizing the utilitarian or egalitarian social welfare has been studied in various settings, e.g., in fractional [5] and additively separable Hedonic Games [4]. To the best of our knowledge, these objectives have not been considered in the context of anonymous Hedonic Games.

In our paper, we show the tractability of computing a wonderful partition for instances with interval approvals as formulated by Woeginger [31]. To this aim, we provide a dynamic program that relies on the approach used in [21] to solve a capacitated rectangle stabbing problem from computational geometry. Here, the goal is to stab a set of rectangles with a minimum subset of a given set of lines, each intersecting (i.e., potentially *stabbing*) some of the rectangles. Each line has a maximum number of rectangles it can stab, and to stab all rectangles, one is allowed to use multiple copies of each line.

We also consider variants with single-peaked cost functions of the agents. Single-peaked preferences date back to Black [12]. Such preferences are a common theme in the Economics and Game Theory literature. In particular, they play a prominent role in different fields such as Hedonic Diversity Games [16, 13], Schelling Games [11, 22], and in various works on voting and social choice [30, 32, 10, 19, 15].

1.2 Model

The hiking problem as formulated by Gerhard Woeginger is a special case of anonymous Hedonic Games with approval-based preferences. In an anonymous Hedonic Game with approval-based preferences we are given a set N of n agents to be partitioned into coalitions. Each agent $i \in N$ reports an approval set $S_i \subseteq [n]$ representing the approved group sizes for agent i. In particular, agent i wants to be in some group of size s such that $s \in S_i$. An approval set S_i is said to be an *interval* if $S_i = \{\ell_i, \ell_i + 1, \dots, r_i\}$, for some $1 \le \ell_i \le r_i \le n$. The agents have to be partitioned into coalitions, i.e., subsets of the agent set. This induces a *partition* π of the set of agents N. We denote by $\pi(i)$ the coalition agent i belongs to in the partition π. We follow the notation in Woeginger's survey paper [31] and call a partition π *wonderful* if each agent approves of the size of its coalition in π, i.e., for each agent i, $|\pi(i)| \in S_i$. This leads to the following natural computational problem called Wonderful-Partition.

Wonderful-Partition
Input: A set N of agents and size approval sets $(S_i)_{i \in N}$.
Problem: Decide whether there exists a wonderful partition of the agents. If yes, compute one.

Woeginger's Hiking Problem is Wonderful-Partition with interval approval sets, which we formally define as follows.

Wonderful-Partition-Intervals (Hiking)
Input: A set N of agents and for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \ldots, r_i\}$.
Problem: Decide whether there exists a wonderful partition of the agents. If yes, compute one.

We also consider natural extensions of the hiking problem, to be introduced later.

1.3 Our Contribution

We solve Woeginger's Hiking Problem in the affirmative by giving an $O(n^5)$-algorithm that computes a wonderful partition for an instance with n agents that each have interval approval sets (see Figure 2 (a)), if such a partition exists. For this we use a dynamic programming approach for a rectangle stabbing problem from computational geometry. Moreover, we extend this approach to achieve an $O(n^5)$-algorithm for computing the minimum set of hikers that have to be excluded from the hike, in order for a wonderful partition to become possible. We also give an $O(n^7)$-algorithm for deciding if a wonderful partition exists if exactly x hikers are excluded. This is then used to derive an $O(n^7 \log n)$-algorithm for finding a partition that maximizes the number of hikers that approve of their assigned group size. These approaches can also be extended to a setting where hikers have weights.

All these positive results hold for the case where the agents have interval approval sets (Figure 2 (a)). The special case where every hiker only approves of one group size, i.e., approval intervals of size 1, can be solved efficiently by checking if for all i the number of agents approving only of size i is divisible by i. In contrast to this, it was already known that

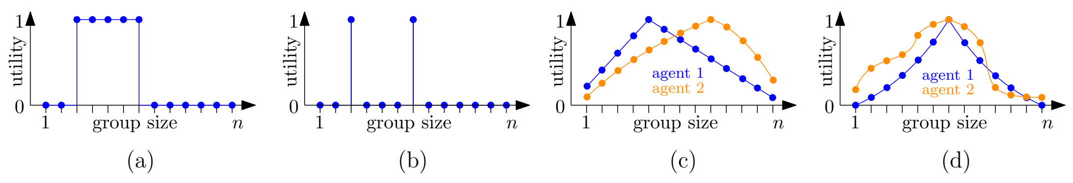

Figure 2 Examples of utility functions considered in this paper. (a) interval approval sets, (b) non-interval approval sets with two approved group sizes, (c) utility functions are single-peaked and all agents with the same peak have the same utility function (under some additional mild technical assumptions), (d) individual single-peaked utility functions for all agents (as shown, the functions may differ even if they have the same peak).

the problem is NP-hard even if each attendee has at most two different approved group sizes that need not form an interval [17] (see Figure 2 (b)). One can modify the original proof to establish hardness even for the case where each attendee approves of *exactly* two group sizes. For the sake of completeness and to improve readability, we show the details in Appendix B. Our presentation hinges on an elegant connection to a graph orientation problem that has not been used in the previous proof.

We complete the picture by considering agents with single-peaked preferences over group sizes. In this setting an agent has a cost that is a function of its assigned group size and its ideal group size. If all agents incur a cost that is given by a fixed function that depends on their peak (but different peak values are allowed, see Figure 2 (c)) and if this function satisfies a mild technical condition, we compute a partition that minimizes the social cost in time $O(n^2(\alpha+1))$, if at most α agents can be excluded from the hike. This holds for the utilitarian setting, i.e., minimizing the sum of the agents' costs, and also for the egalitarian setting, where the maximum agent cost is to be minimized. Finally, we prove polynomial time equivalence of the wonderful partition problem and the problem of minimizing the egalitarian social cost in (general) anonymous Hedonic Games. We use this result to show that even for the setting where each agent has its own single-peaked cost function (see Figure 2 (d)), a partition that minimizes the egalitarian social cost can be computed in $O(n^5 \log n)$.

2 Efficient Algorithm for Woeginger's Hiking Problem, and Extensions

We prove that WONDERFUL-PARTITION-INTERVALS can be solved in polynomial time by casting it as a version of capacitated rectangle stabbing that is efficiently solvable via an elegant dynamic programming approach introduced by Even, Levi, Rawitz, Schieber, Shahar, and Srividenko [21]. In the capacitated rectangle stabbing problem, we are given axis-parallel rectangles and a set of axis-parallel lines. The goal is to find a minimum number of lines that intersect and 'stab' all rectangles, or, in a version where lines have costs, lines with minimum total cost with the same property. Each line has a maximum number of intersected rectangles that it can 'stab', and, to stab all rectangles, one is allowed to use multiple copies of each line. Woeginger's Hiking Problem can be seen as a special case of the one-dimensional capacitated rectangle stabbing problem with costs, where the agents' intervals correspond to one-dimensional rectangles and stabbing rectangles at some integer position i costs i and corresponds to creating a group of size i in which we include the agents corresponding to the stabbed rectangles. Solutions with total cost n (which is also a lower bound on the cost) then correspond to those where all groups are full. Building on the approach of [21] applied to this case, we give a simpler $\mathcal{O}(n^5)$ dynamic program that is specifically tailored to our needs. Note that this reduction from our partition problem to a version of rectangle stabbing draws a powerful new connection between the problem types that is not limited to one specific variant of the partition problem, but also holds more generally; e.g., it can be adapted to the variant where there can be at most one group of each size.

In the following, we use the central observations of Even, Levi, Rawitz, Schieber, Shahar, and Srividenko [21] to derive a dynamic programming solution to Woeginger's Hiking problem, i.e., to WONDERFUL-PARTITION-INTERVALS. Its definition and analysis will then also power our results on the more demanding problem variants discussed in the introduction.

2.1 Dynamic Program for Woeginger's Hiking Problem

Consider an instance of WONDERFUL-PARTITION-INTERVALS, i.e., a set N of n agents, without loss of generality $N = [n]$, and n pairs $(\ell_i, r_i)_{i\in N}$ such that agent $i \in [n]$ approves of sizes $S_i = \{\ell_i, \dots, r_i\}$. We are interested in checking whether there exists a wonderful partition of the agents; i.e., one where every agent approves of their coalition size. For consistent tie-breaking reasons, throughout this section we write $i \prec j$ for two distinct agents $i, j \in [n]$ if either $r_i < r_j$, or $r_i = r_j$ and $i < j$. Note that $\prec$ is a well-defined strict linear order and, without loss of generality, assume that agents are ordered so that $1 \prec \dots \prec n$.

As the main technical ingredient of our approach, given a subset $N' \subseteq N$ of agents, we say that a wonderful partition π of N' is *earliest-due-date* if for any two agents $i \prec j$ it does not hold that $\ell_i \leq |\pi(j)| < |\pi(i)|$. This terminology is borrowed from the scheduling literature, and based on the following well-known observation: let us view the agents' allowed intervals of coalition sizes on an axis labeled with the natural numbers $1, \dots, n$, and consider an arbitrary collection of coalition sizes we might decide on using, in non-decreasing order of coalition size. Then, going in this order, it is always safe to include those agents first whose right interval endpoints are the smallest: in said order of coalitions, they are the ones that stop being servable first, making an earliest-due-date approach impose the least restrictions on later assignments. In other words, whenever a wonderful partition into the given coalition sizes exists, there also exists one that is earliest-due-date. We prove this formally below for completeness. Note that, to make the process well-defined, instead of comparing agents by right endpoints, we compare them by $\prec$.

▶ **Lemma 1.** *If a subset $N' \subseteq N$ of agents admits a wonderful partition, then it admits an earliest-due-date wonderful partition.*

Proof. Consider a wonderful partition π of N'. If it satisfies the required property, then we are done, otherwise, consider $i, j \in N'$ such that $i \prec j$ and $\ell_i \leq |\pi(j)| < |\pi(i)|$. These conditions imply that $|\pi(j)| \in [\ell_i, r_i]$ and $|\pi(i)| \in [\ell_j, r_j]$. Hence, one can construct a new wonderful partition π' from π by exchanging the groups of agents i and j; i.e., $\pi'(i) = (\pi(j) \setminus \{j\}) \cup \{i\}$ and $\pi'(j) = (\pi(i) \setminus \{i\}) \cup \{j\}$. Subsequently, set $\pi \leftarrow \pi'$ and repeat the same procedure until the required condition is satisfied. To complete the proof, we need to show that this is eventually the case. To do so, since $1 \prec \dots \prec n$, each exchange strictly increases the sequence $(|\pi(n)|, \dots, |\pi(1)|)$ lexicographically. Because this sequence is bounded from above by $(n, \dots, n)$, the process eventually ends. ◀

Hence, from now on, we will only seek earliest-due-date wonderful partitions. This crucial restriction, which we have shown to be without loss of generality, will allow us to bootstrap a dynamic programming algorithm that determines a wonderful partition if one exists.

To construct our algorithm, we first show that earliest-due-date partitions admit an attractive recursive decomposition. Consider an earliest-due-date wonderful partition π of the agents (if any exist), and consider the size of the coalition that agent n is a part of; i.e., $|\pi(n)|$. Moreover, consider an arbitrary agent $i \neq n$. Agent i is part of a coalition of size $|\pi(i)|$. Because π is earliest-due-date, by definition it can not be the case that $\ell_i \leq |\pi(n)| < |\pi(i)|$, so either $|\pi(n)| < \ell_i$ or $|\pi(i)| \leq |\pi(n)|$ holds.

▶ **Lemma 2.** *Consider an earliest-due-date wonderful partition π of N. Partition the agents as $N = N_- \cup N_+ \cup \{n\}$, where $N_- = \{i \in N \setminus \{n\} \mid \ell_i \leq |\pi(n)|\}$ and $N_+ = \{i \in N \setminus \{n\} \mid \ell_i > |\pi(n)|\}$, then it holds that:*

1. *For all $i \in N_-$ we have $|\pi(i)| \leq |\pi(n)|$;*
2. *For all $i \in N_+$ we have $|\pi(i)| > |\pi(n)|$.*

Proof. For (2) note that any $i \in N_+$ by definition satisfies $\ell_i > |\pi(n)|$. Since we have the requirement that $|\pi(i)| \in [\ell_i, r_i]$, this means that $|\pi(i)| \geq \ell_i > |\pi(n)|$, as desired.

For (1), consider some $i \in N_-$. By definition, we have that $\ell_i \leq |\pi(n)|$. Assume for a contradiction that $|\pi(i)| > |\pi(n)|$. This implies that $\ell_i \leq |\pi(n)| < |\pi(i)|$. Since $r_i \leq r_n$ and by the sorting criterion we also have $i \prec n$, which contradicts that π is earliest-due-date. ◀

Notably, the same result holds true if we work with a subset of the agents $N' \subsetneq N$ and n is replaced with the agent $a \in N'$ with maximum r_a.

Armed as such, to build intuition, Lemma 2 tells us that no two agents $a_- \in N_-$ and $a_+ \in N_+$ can go into the same group, so a first attempt at a recursive algorithm looking for a wonderful partition of N, to be later optimized by memorization/dynamic programming, would proceed as follows: start with $N' = N$; at each step, identify the agent $a \in N'$ maximizing r_a and exhaust over all possibilities for $|\pi(a)|$; for each one, write $N' = N'_- \cup N'_+ \cup \{a\}$ as defined in Lemma 2 and recurse with N'_- and N'_+. We will specify the details of the process such that, if the recursive calls yield wonderful partitions of N'_- and N'_+, a wonderful partition of N can also be constructed by incorporating agent a into them. Of course, the last step of reasoning is incorrect, since one needs to make sure that a group of size $|\pi(a)|$ with one space available indeed exists, and this information needs to be somehow propagated across recursive calls. Moreover, it is unclear what the number of sets N' reachable by the recursion is. This has to be polynomially bounded so that memorization/dynamic programming leads to a polynomial-time algorithm.

Let us first address the second issue outlined above, namely the size of the state space. This turns out to be relatively simple: define $N(x_1, x_2, i) = \{j \in N \mid j \leq i \text{ and } \ell_j \in [x_1, x_2]\}$. Then, we can adapt the recursive approach: instead of storing N' explicitly, start with $x_1 = 1$ and $x_2 = n$, as well as $i = n$, from which implicitly $N' = N(x_1, x_2, i)$. At each step we will first check whether i, which by the sorting criterion is the largest agent maximizing r_i, is in $N' = N(x_1, x_2, i)$; i.e., whether $\ell_i \in [x_1, x_2]$. If not, then $N(x_1, x_2, i) = N(x_1, x_2, i-1)$ and we simply recurse with the same x_1, x_2 and $i' = i - 1$. Otherwise, i is the largest agent in N' maximizing r_i, so we can exhaust as before over all possible values for $|\pi(i)|$ and perform two recursive calls in each case: one with $(x'_1, x'_2, i') = (x_1, |\pi(i)|, i-1)$ and one with $(x'_1, x'_2, i') = (|\pi(i)| + 1, x_2, i-1)$. We have yet to solve the correctness issue, but we have made progress: the state space now consists of triples (x_1, x_2, i) such that $1 \leq x_1 \leq x_2 \leq n$ and $1 \leq i \leq n$, which there are $O(n^3)$ of.

Let us now turn our attention to ensuring correctness. To do so, we need to investigate more closely how to ensure that the group of size $|\pi(i)|$ of agent i (recall that we only exhaust over its size, not over which agents are in it) exists and is used to its full capacity in the solution constructed by the recursion. There is one crucial guarantee given by Lemma 2 that we have so far not exploited. Namely, the recursive call with $(x_1, |\pi(i)|, i-1)$ should only use groups of sizes $s \in [x_1, |\pi(i)|]$ and the one with $(x'_1, x'_2, i') = (|\pi(i)| + 1, x_2, i-1)$ should only use groups of sizes $s \in [|\pi(i)| + 1, x_2]$. In general, state (x_1, x_2, i) should only consider groups of sizes $s \in [x_1, x_2]$. This can be ensured by requiring that the exhausted value for $|\pi(i)|$ additionally satisfies $x_1 \leq |\pi(i)| \leq x_2$. This small, seemingly inconsequential, refinement of the approach will ultimately allow us to propagate information about incomplete groups across states in the recursion. Before describing how this is done in general, to build more intuition, let us consider the first level of the recursion, namely $(x_1, x_2, i) = (1, n, n)$. At this level, the algorithm exhausts over the possible values for $|\pi(n)| \in [\ell_n, r_n]$. For each possibility, in the ensuing recursive call with $(x'_1, x'_2, i') = (|\pi(n)| + 1, n, n-1)$ all used groups will be of size at least $|\pi(n)| + 1 > |\pi(n)|$ so nothing special needs to be done in this case since those agents can not be part of agent n's group. However, in the other call, having $(x'_1, x'_2, i') = (1, |\pi(n)|, n-1)$ some $|\pi(n)| - 1$ other agents will need to share a group of size $|\pi(n)|$ with agent n, and this is not yet modeled by our approach. To model this effect across recursive calls, we introduce a fourth variable $0 \leq k < x_2$ to the state of our recursive algorithm; i.e., each state is now of the form (x_1, x_2, i, k). This variable intuitively signifies that there exists (from upward in the recursion) an incomplete group, currently of size k, whose final size should be x_2. With this setup, the starting top-level call will now be with $(x_1, x_2, i, k) = (1, n, n, 0)$. For each value of $|\pi(n)|$, the two resultant recursive calls will be with $(x_1', x_2', i', k') = (1, |\pi(n)|, n-1, 1)$[3] and $(x_1', x_2', i', k') = (|\pi(n)|+1, n, n-1, 0)$. Note how the former call has $k' = 1$, signifying that we just "opened a new (incomplete) group of size one, whose final size should be $x_2' = |\pi(n)|$" The fundamental reason why such an approach can work is that in any node of the recursion tree, there is at most a single incomplete group to keep track of. Note that this fact crucially depends on each call (x_1, x_2, i, k) only considering partitions into groups of sizes between x_1 and x_2. We still need to describe the transitions for general calls (x_1, x_2, i, k) given the newly added parameter k. The formal details will following below, but in rough lines, the call for N_- creates a new group; i.e., $k' = 1$; while the call for N_+ keeps the currently open one; i.e., $k' = k$; the exception comes when $|\pi(i)| = x_2$, in which case the call for N_- adds to the same group, possibly closing the group; i.e., $k' = (k+1) \pmod{x_2}$; and no call for N_+ is generated.

▶ **Theorem 3.** *Deciding whether a wonderful partition exists and computing one if so can be achieved in $O(n^5)$ time.*

Proof. We show how to solve the decision version. Constructing a wonderful partition for yes-instances can be subsequently achieved by standard techniques. We proceed by dynamic programming (DP). Define the Boolean DP array $dp[x_1, x_2, i, k]$ for $1 \leq x_1 \leq x_2 \leq n$, $0 \leq i \leq n$ and $0 \leq k < x_2$, with the meaning $dp[x_1, x_2, i, k] = 1$ if and only if there exists a wonderful partition of agents in $N(x_1, x_2, i)$ using only groups of sizes in $[x_1, x_2]$ and assuming that we start with an incomplete group of current size k which has to have final size x_2. Naturally, if $k = 0$ this means starting with no partially filled group. The final answer will be available at the end in $dp[1, n, n, 0]$. To compute the DP table, we use the following recurrence relations and base cases (using a hyphen as stand-in for any group size):

1. $dp[-, -, 0, 0] = 1$;
2. $dp[-, -, 0, k] = 0$ if $k \neq 0$;
3. $dp[x_1, x_2, i, k] = dp[x_1, x_2, i-1, k]$ if $\ell_i \notin [x_1, x_2]$;
4. $dp[x_1, x_2, i, k] = \bigvee_{x=\ell_i}^{\min(x_2, r_i)} F_x$ if $\ell_i \in [x_1, x_2]$, where

$$F_x := \begin{cases} dp[x_1, x_2, i-1, (k+1) \bmod x_2] & x = x_2 \\ dp[x_1, x, i-1, 1 \bmod x] \wedge dp[x+1, x_2, i-1, k] & x < x_2. \end{cases}$$

The first three cases are immediate from the definition of the DP. For the last case, we iterate over $x \in [\ell_i, \max(x_2, r_i)]$ which is agent i's group size. There are two cases: if $x = x_2$, then we put i into the group of current size k and final size x_2 that we assumed to have at our disposal; this increases the size of the group by one, or completes it if $k = x_2 - 1$; otherwise, $x < x_2$, and we recurse into partitioning agents in $N(x_1, x, i-1)$ into groups of sizes between x_1 and x, and $N(x+1, x_2, i-1)$ into groups of sizes between $x+1$ and x_2. For the first call, this generates a new group of size 1 (unless $x = 1$), while for the latter, this keeps the previously open group of current size k and final size x_2 that we assumed to have.

[3] Strictly speaking, this should be $(x_1', x_2', i', k') = (1, |\pi(n)|, n-1, 1 \bmod |\pi(n)|)$. We omit this detail from the higher-level exposition to improve readability.

To compute the DP table in an acyclic fashion, it suffices to iterate through i in ascending order. The complexity of the approach is $O(n^5)$ because there are $O(n^4)$ states and computing the value for states of type (4) requires iterating through $O(n)$ values of x. ◀

2.2 Extensions

Using a similar DP approach, we can solve the following natural extension.

Hiking-Min-Delete
Input: A set N of agents and for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \dots, r_i\}$.
Problem: Compute a set $N' \subseteq N$ of minimum size such that $N \setminus N'$ has a wonderful partition. Output N' and a wonderful partition of $N \setminus N'$.

The approach relies on essentially the same recursive reasoning as before, except that we now also consider the possibility of "ignoring" an agent i and hence recursing with $(x_1', x_2', i', k') = (x_1, x_2, i-1, k)$. Moreover, instead of making the recursion return whether a wonderful partition is possible or not, we make it return the minimum number of agents that need to be removed so that this is possible. In light of this, the "ignore i" recursive call incurs a cost of 1 removed agent. The details are formalized in the following.

▶ **Theorem 4.** *Hiking-Min-Delete is solvable in $O(n^5)$ time.*

Proof. We use a similar approach as in the proof of Theorem 3. As before, it suffices to show how to compute the minimum size of N', as computing such a set N' and a corresponding wonderful partition for $N \setminus N'$ follow by standard techniques. Define the integer-valued DP array $dp[x_1, x_2, i, k]$ for $1 \leq x_1 \leq x_2 \leq n$, $0 \leq i \leq n$ and $0 \leq k < x_2$, with the meaning that $dp[x_1, x_2, i, k]$ contains the minimum number of agents which need to be removed from $N(x_1, x_2, i)$ so that the remaining agents admit a wonderful partition into groups of sizes in $[x_1, x_2]$ assuming that we start with an incomplete group of current size k which has to have final size x_2. Note that, in contrast to the previous DP, the value 0 corresponds to removing no agents, which is the best possible outcome, while previously it corresponded to the need for removing at least one agent. The final answer will be available at the end in $dp[1, n, n, 0]$. To compute the DP table, we use the following recurrence relations and base cases:

1. $dp[-, -, 0, 0] = 0$;
2. $dp[-, -, 0, k] = \infty$ if $k \neq 0$;
3. $dp[x_1, x_2, i, k] = dp[x_1, x_2, i-1, k]$ if $\ell_i \notin [x_1, x_2]$;
4. $dp[x_1, x_2, i, k] = \min\{1 + dp[x_1, x_2, i-1, k], X\}$ if $\ell_i \in [x_1, x_2]$, where

$$X := \min\{F_x \mid \ell_i \leq x \leq \min(x_2, r_i)\}$$

and

$$F_x := \begin{cases} dp[x_1, x_2, i-1, (k+1) \bmod x_2] & x = x_2 \\ dp[x_1, x, i-1, 1 \bmod x] + dp[x+1, x_2, i-1, k] & x < x_2. \end{cases}$$

The reasoning stays largely the same as before, with the only difference being the new $1 + dp[x_1, x_2, i-1, k]$ term, which accounts for discarding agent i and incurring a cost of 1 corresponding to removing an agent. ◀

Another subtly distinct natural variant of Woeginger's Hiking Problem is the following:

Hiking-Max-Satisfied
Input: A set N of agents and for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \dots, r_i\}$.
Problem: Compute a partition π of N maximizing the number of agents approving of their coalition sizes.

Indeed, Hiking-Max-Satisfied and Hiking-Min-Delete are related, in that if N' is a minimum-size set of agents such that $N \setminus N'$ has a wonderful partition, then it is also possible to satisfy at least $n - |N'|$ agents in a partition of N. This is because agents in N' can be put together in a group in tandem to a wonderful partition of $N \setminus N'$ to get a partition of N with at least $n - |N'|$ satisfied agents. However, it might be possible to satisfy more than $n - |N'|$ agents if unsatisfied agents do not all go into the same group.

Hence, solving this variant of the problem introduces new challenges that require further insight. Let us fix a number k and ask whether it is possible to satisfy at least $|N| - k$ agents. If the answer is affirmative, we also want a partition achieving this. To solve Hiking-Max-Satisfied, we will binary search for the smallest $0 \leq k \leq |N|$ for which the answer is affirmative, call it k^*, and then recover a partition for k^*. It remains to show how to solve the problem for a fixed value of k. To do so, we need to adjust the angle from which we look at the problem. In particular, instead of looking for a partition satisfying at least k agents, we will look for a size-k subset $N' \subseteq N$ (corresponding to k agents which we do not require to be satisfied) such that $(N \setminus N') \cup D_k$ admits a wonderful partition, where D_k is a set of k dummy agents happy with any group size. Intuitively, k agents are replaced with dummies not minding their group size. Because it is always no worse to remove agents from N in contrast to removing agents from D_k, it is enough to ask to remove exactly k agents from $N \cup D_k$ such that the remaining agents admit a wonderful partition. Checking whether this is possible and finding a corresponding wonderful partition reduces to the following more general problem, which we will show can be solved in polynomial time.

Hiking-x-Delete
Input: A set N of agents, for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \dots, r_i\}$, and also a number $0 \leq x \leq |N|$.
Problem: Compute a set $N' \subseteq N$ of size x such that $N \setminus N'$ has a wonderful partition (or report impossibility). Output N' and a wonderful partition of $N \setminus N'$.

We now show how to solve Hiking-x-Delete in polynomial time. We can once again try to attack the problem recursively with our usual state (x_1, x_2, i, k). Just like we did for Hiking-Min-Delete, we will have recursive calls for ignoring an agent; i.e., adding it to the set N' of removed agents.[4] In order to ensure that exactly x agents are removed, we add another variable r to the state of the recursion: (x_1, x_2, i, k, r), where r is how many agents we want to remove. The top-level call will be invoked with $r = x$. The recursive approach for Hiking-Min-Delete now translates relatively swiftly to the new setting. The main difference is the case where a state of the form $(x_1, x_2, i, k, -)$ with $\ell_i \in [x_1, x_2]$ performs two recursive calls to $(x_1, x, i-1, 1, -)$ and $(x+1, x_2, i-1, k, -)$. In particular, say the

[4] Note that previously we used N' to denote $N(x_1, x_2, i)$ when discussing the recursive algorithm for the original hiking problem. We do not keep this notation here.

state is (x_1, x_2, i, k, r), then what should be the values r' and r'' for the two recursive calls? Intuitively, there is no fixed answer, since it could be that we remove more agents in the first or in the second call. In fact, we have full freedom over how to split the r removals across the two calls as long as $r' + r'' = r$. Hence, we will iterate over all options $0 \leq r' \leq r$ and set $r'' = r - r'$, and in each case call recursively with $(x_1, x, i-1, 1, r')$ and $(x+1, x_2, i-1, k, r'')$.

▶ **Theorem 5.** *Hiking-x-Delete is solvable in $O(n^7)$ time.*

Proof. As before, it suffices to check feasibility, a solution can then be recovered using standard techniques. Define the Boolean DP array $dp[x_1, x_2, i, k, r]$ for $1 \leq x_1 \leq x_2 \leq n$, $0 \leq i \leq n$, $0 \leq k < x_2$, and $0 \leq r \leq x$ with the meaning that $dp[x_1, x_2, i, k, r] = 1$ if and only if there exist r agents that can be removed from $N(x_1, x_2, i)$ such that the remaining agents admit a wonderful partition into groups of sizes in $[x_1, x_2]$ assuming that we start with an incomplete group of current size k which has to have final size x_2. At the end, $dp[1, n, n, 0, x]$ will be 1 if and only if it is possible to remove x agents from N such that the remaining agents admit a wonderful partition. To compute the DP table, we use the following:

1. $dp[-,-,0,0,0] = 1$;
2. $dp[-,-,0,k,r] = 0$ if $k \neq 0$ or $r \neq 0$;
3. $dp[x_1, x_2, i, k, r] = dp[x_1, x_2, i-1, k, r]$ if $\ell_i \notin [x_1, x_2]$;
4. $dp[x_1, x_2, i, k, r] = dp[x_1, x_2, i-1, k, r-1, d] \vee X$[5] if $\ell_i \in [x_1, x_2]$, where $X := \bigvee_{x=\ell_i}^{\min(x_2, r_i)} F_x$ and

$$F_x := \begin{cases} dp[x_1, x_2, i-1, (k+1) \bmod x_2, r] & x = x_2 \\ \bigvee_{r'=0}^{r} \left(dp[x_1, x, i-1, 1 \bmod x, r'] \wedge dp[x+1, x_2, i-1, k, r-r']\right) & x < x_2. \end{cases}$$

We explain the four cases separately: for cases (1) and (2) we have $i = 0$; i.e., the set of yet-to-be-considered agents is empty, so $k = r = 0$ is necessary and sufficient for the table to store a 1, signifying feasibility. As before, case (3) straightforwardly ignores an agent that is not part of the current set of active agents.

The more interesting case is case (4). First, the $dp[x_1, x_2, i-1, k, r-1]$ term corresponds to removing an agent, and it only applies to $r \geq 1$, as in the footnote. This decreases r by 1 since we have just removed an agent. The $X := \bigvee_{x=\ell_i}^{\min(x_2, r_i)} F_x$ term is more involved. The selection of $\ell_i \leq x \leq \min(x_2, r_i)$ corresponds to selecting the size of the group that agent i will be part of. For $x = x_2$, the analysis is similar to our previous DPs. For $x < x_2$, on the other hand, we moreover split the r agent removals into r' removals for the first recursive call and $r'' = r - r'$ for the second. Whether both calls are successful is represented by the expression $dp[x_1, x, i-1, 1 \bmod x, r'] \wedge dp[x+1, x_2, i-1, k, r'']$.

As before, to compute the DP table in an acyclic fashion, it suffices to iterate through i in ascending order. The complexity is $O(n^7)$ because there are $O(n^5)$ states and computing the value for states of type (4) requires iterating through $O(n^2)$ values for (x, r). ◀

Using the above binary search approach we get a solution for Hiking-Max-Satisfied that runs only a $O(\log n)$ factor slower than the runtime for Hiking-x-Delete.

▶ **Theorem 6.** *Hiking-Max-Satisfied is solvable in $O(n^7 \log n)$ time.*

[5] Only X for $r = 0$ as otherwise we would be referring to the invalid value $r = -1$.

2.3 Further Weighted Extensions

If not all agents can be satisfied, Hiking-Min-Delete and Hiking-Max-Satisfied provide two ways of implementing a compromise. However, both treat unsatisfied/deleted agents equally. In certain settings, it might be more desirable to take into account the different *entitlements* of the agents; i.e., one agent might have been dissatisfied with their group size during the previous edition of the workshop, or another agent might be the senior invited speaker. One modelling option is to assign a weight w_i to each agent $i \in N$ and weigh the dissatisfied/deleted agents accordingly leading to the following variants:

Hiking-Min-Delete-Weighted
Input: A set N of agents and for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \dots, r_i\}$. Moreover, for each agent $i \in N$, a number $w_i \in \mathbb{R}_{\geq 0}$.
Problem: Compute a set $N' \subseteq N$ minimizing $\sum_{i \in N'} w_i$ such that $N \setminus N'$ has a wonderful partition. Output N' and a wonderful partition of $N \setminus N'$.

Hiking-Max-Satisfied-Weighted
Input: A set N of agents and for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \dots, r_i\}$. Moreover, for each agent $i \in N$, a number $w_i \in \mathbb{R}_{\geq 0}$.
Problem: Compute a partition π of the agents such that if N_π is the set of agents approving of their coalition sizes in π, then $\sum_{i \in N_\pi} w_i$ is maximized.

Our dynamic programs, with minor modifications which we sketch next, can also be used to solve the weighted variants. We begin with Hiking-Min-Delete-Weighted.

▶ **Theorem 7.** *Hiking-Min-Delete-Weighted is solvable in $O(n^5)$ time.*

Proof Sketch. In the proof of Theorem 4, we defined the DP as follows: "$dp[x_1, x_2, i, k]$ contains the minimum number of agents which need to be removed from $N(x_1, x_2, i)$ so that the remaining agents admit a wonderful partition into groups of sizes in $[x_1, x_2]$ assuming that we start with an incomplete group of current size k which has to have final size x_2". To handle weights, this is replaced by "$dp[x_1, x_2, i, k]$ contains the minimum total weight of agents which need to be removed from $N(x_1, x_2, i)$ so that the remaining agents admit a wonderful partition into groups of sizes in $[x_1, x_2]$ assuming that we start with an incomplete group of current size k which has to have final size x_2". The rest of the proof stays the same, with the minor adaptation that the fourth recurrence relation accordingly becomes $dp[x_1, x_2, i, k] = \min\{w_i + dp[x_1, x_2, i-1, k], X\}$, where previously $w_i = 1$ has been used. ◀

To get a similar result for Hiking-Max-Satisfied-Weighted following our previous proof outline, we first define a weighted analogue of Hiking-x-Delete, as follows.

Hiking-x-Delete-Min-Weight
Input: A set N of agents, for each agent $i \in N$ two numbers $\ell_i \leq r_i$ such that $S_i = \{\ell_i, \dots, r_i\}$, and also a number $0 \leq x \leq |N|$. Moreover, for each agent $i \in N$, a number $w_i \in \mathbb{R}_{\geq 0}$.
Problem: Compute a set $N' \subseteq N$ of size x minimizing $\sum_{i \in N'} w_i$ such that $N \setminus N'$ has a wonderful partition (or report impossibility). Output N' and a wonderful partition of $N \setminus N'$.

▶ **Theorem 8.** *Hiking-x-Delete-Min-Weight is solvable in $O(n^7)$ time.*

Proof Sketch. In the proof of Theorem 5 for the unweighted version, we defined a boolean DP as follows: "$dp[x_1, x_2, i, k, r] = 1$ if and only if there exist r agents that can be removed from $N(x_1, x_2, i)$ such that the remaining agents admit a wonderful partition into groups of sizes in $[x_1, x_2]$ assuming that we start with an incomplete group of current size k which has to have final size x_2". For the current problem, we want the states to signal not only possibility/impossibility but also what is the minimum total weight of those r removed agents. Hence, we replace this definition by "$dp[x_1, x_2, i, k, r]$ is the minimum total weight of r agents that can be removed from $N(x_1, x_2, i)$ such that [...] (or ∞ if impossible)". The rest of the reasoning stays analogous with minor changes: the values $0, 1$ in the base cases become $\infty, 0$, disjunctions ($\vee$) are replaced by "min" and conjunctions ($\wedge$) by $+$. Finally, the fourth recurrence relation becomes $dp[x_1, x_2, i, k, r] = \min\{w_i + dp[x_1, x_2, i-1, k, r-1, d], X\}$. ◀

We make use of Theorem 8 to show the following.

▶ **Theorem 9.** *Hiking-Max-Satisfied-Weighted is solvable in $O(n^8)$ time.*

Proof Sketch. In the proof of Theorem 6 for the unweighted case, we were looking for the largest k such that there exists a size-k subset $N' \subseteq N$ (corresponding to k agents which we do not require to be satisfied) such that $(N \setminus N') \cup D_k$ admits a wonderful partition, where D_k was a set of k dummy agents happy with any group size. We then argued that we could relax to asking for a size-k subset $N' \subseteq (N \cup D_k)$ such that $(N \cup D_k) \setminus N'$ admits a wonderful partition, which can be done using our polynomial-time algorithm for Hiking-x-Delete-Min in Theorem 5. This time, we follow a similar approach, except without binary search: we will try out all values $0 \leq k \leq n$ and ask for a size-k subset $N' \subseteq (N \cup D_k)$ such that $(N \cup D_k) \setminus N'$ admits a wonderful partition. For a fixed k, out of all abiding N', we want one minimizing $\sum_{i \in N'} w_i$. Such an N' can be computed using our $O(n^7)$ algorithm for Hiking-x-Delete-Min-Weight in Theorem 8. We do this for all values $0 \leq k \leq n$ and take the minimum-weight solution, adding an extra $O(n)$ factor, so the overall complexity is $O(n^8)$. ◀

3 Single-Peaked Preferences Over Group Sizes

We extend the binary version of the hiking problem by considering single-peaked preferences over group sizes. We assume that each agent i has an ideal group size s_i and the cost of agent i if placed in a group of size s is given by a cost function dependent on s_i and s.

Given these ingredients, minimizing the social cost can be done in two variants: a utilitarian variant and an egalitarian variant. In the utilitarian variant, the goal is to minimize the total cost of the agents, while in the egalitarian version we want the cost of the agent having the highest cost to be as low as possible, i.e., we replace summation with maximum. We are also interested in a variation of the problem where the hike organizers consider it reasonable to exclude at most α of the agents from the hike, the goal becoming to select the agents to remove and then to organize a hike with best social cost among the remaining agents.[6] Note that we assume that each excluded agent has a cost of 0. This is different from assuming that excluded agents have to hike alone, which is covered by the case $\alpha = 0$.

More formally, we assume that each agent $i \in N$ announces an ideal coalition size s_i; i.e., agent i would be most happy when belonging to a coalition of size s_i. Moreover, given

[6] The original problem can be seen to be the $\alpha = 0$ case.

some cost function $cost : N^2 \to \mathbb{R}$ agent i incurs a cost of $cost(s_i, s)$ if placed in a coalition of size $s \in [n]$ and a cost equal to 0 if not participating in the hike. We assume c to be monotone, i.e., $cost(s_i, s) \leq cost(s_i, s')$ if $s_i \leq s \leq s'$ or $s' \leq s \leq s_i$. Notice that since agent i incurs a disutility equal to $cost(s_i, s)$, where s_i is the most preferred size of i, the monotonicity condition on $cost(\cdot, \cdot)$ implies that the preferences of agents are single-peaked w.r.t. the natural ordering. We refer to the next section for a formal definition. Given a partition π, we recall that $\pi(i)$ denotes the coalition of agent i; if i is not participating in the hike we write $\pi(i) = \perp$. Furthermore, by slight abuse of notation, for an agent i we write $i \in \pi$ to indicate that agent i takes part in the hike; i.e., $\pi(i) \neq \perp$.

The utilitarian social cost of a partition π is given by $cost(\pi) = \sum_{i \in \pi} cost(s_i, |\pi(i)|)$ while the egalitarian social cost is given by $cost(\pi) = \max_{i \in \pi} cost(s_i, |\pi(i)|)$. The goal is, therefore, to find a partition minimizing the social cost under the constraint $|\{i \mid \pi(i) = \perp\}| \leq \alpha$, or equivalently, $|\{i \mid i \in \pi\}| \geq n - \alpha$, where α is the maximum number of agents that are allowed to not participate in the hike. Without loss of generality, we assume that $s_1 \leq \ldots \leq s_n$.

We begin by proving two structural properties of optimal solutions that will allow us to greatly reduce the space of solutions that have to be considered, hence enabling us later to give efficient dynamic programming algorithms computing optimal partitions for both the utilitarian and the egalitarian settings. For the utilitarian social cost we need a mild assumption on the cost function *cost*.

▶ **Definition 10.** *A function* $cost : N^2 \to \mathbb{R}$ *fulfills the* quadrangle inequality *if and only if*

$$cost(a, c) + cost(b, d) \leq cost(a, d) + cost(b, c) \text{ for all } a \leq b \leq c \leq d.$$

Analogously, it fulfills the reverse quadrangle inequality *if and only if*

$$cost(a, c) + cost(b, d) \leq cost(a, d) + cost(b, c) \text{ for all } a \geq b \geq c \geq d.$$

Note that quadrangle inequality and reverse quadrangle inequality are equivalent if *cost* is symmetric, i.e. $cost(a, b) = cost(b, a)$. Moreover, notice that if $cost(\cdot, \cdot)$ is the Euclidean distance on $\mathbb{R}$ or $cost(a, b) = |b - a|^k$ for any $k \geq 1$, then it satisfies both aforementioned quadrangle inequalities. The first observation that our approach will hinge upon is that it is enough to consider *size-monotonic* partitions.

▶ **Definition 11** (Size-Monotonicity). *A partition π is* size-monotonic *if for any two agents $i, j \in \pi$, with $i < j$, it holds that $|\pi(i)| \leq |\pi(j)|$.*

Roughly speaking, there are optimal solutions where all participating agents with lower preferred coalition sizes belong to smaller coalitions than agents with higher preferred sizes. We show this fact in the following, proven in Appendix A.

▶ **Lemma 12.** *The following properties hold:*

(a) *In the utilitarian setting, if $cost(\cdot, \cdot)$ is monotone and fulfills quadrangle inequality and reverse quadrangle inequality, then there exists a size-monotonic optimal solution.*

(b) *In the egalitarian setting, if $cost(\cdot, \cdot)$ is monotone, then there exists a size-monotonic optimal solution.*

Finally, we will show that it is enough to consider size-monotonic partitions which are additionally *compact*. The latter is defined as follows:

▶ **Definition 13** (Compactness). *A coalition C is* compact *if it is of the form $C = \{i, i + 1, \ldots, j\}$ for some $i \leq j$. A solution π is* compact *if all coalitions $C \in \pi$ are compact.*

We now show that we can modify any optimal size-monotonic partition so that it is size-monotonic and compact, as follows. See Appendix A for the proof.

▶ **Lemma 14.** *There exists an optimal partition which is size-monotonic and compact.*

With the above observations, we now know that it is enough to give an efficient algorithm to compute the best size-monotonic and compact solution. We do so in the following. In fact, our algorithm will only directly leverage compactness.[7] To begin, for any two agents $i \leq j$ define $c(i,j)$ to be the social cost induced by agents $i, i+1, \ldots, j$ when forming the coalition $\{i, i+1, \ldots, j\}$. In particular, $c(i,j) = \sum_{k=i}^{j} cost(s_k, j-i+1)$ in the utilitarian case, and similarly with summation replaced by maximum in the egalitarian case. With this definition in place, note that selecting the best compact solution π with at most α agents not taking part in the hike amounts to selecting compact non-intersecting coalitions $C_1, \ldots, C_\ell$ where $C_k = \{a_k, a_k+1, \ldots, b_k-1, b_k\}$, such that $\sum_{k=1}^{\ell}(b_k - a_k + 1) \geq n - \alpha$, and the sum/maximum of $c(a_1, b_1), \ldots, c(a_\ell, b_\ell)$ is minimized. Without loss of generality we can assume that $b_1 < a_2$, $b_2 < a_3$, …, $b_{\ell-1} < a_\ell$, i.e., we assume that the coalitions are sorted by index in increasing order. Before giving the actual algorithm, we note that, to get the best efficiency possible, we will need that the values $c(i,j)$, for all pairs (i,j) with $i \leq j$, can be computed in total time $O(n^2)$ as a preprocessing step. We show this now.

▶ **Lemma 15.** *All values $c(i,j)$ for $i \leq j$ can be computed in total time $O(n^2)$.*

Proof. We will compute the values separately for each value of $j - i$. In particular, for each $0 \leq \ell < n$ we will compute all values $c(i, i+\ell)$ for $1 \leq i \leq n-\ell$ in linear time. To do this, for a fixed ℓ, note the contribution of agent k to the c-values that it counts into is precisely $cost(s_k, \ell+1)$. Therefore, the values $[c(i, i+\ell)]_{1 \leq i \leq n-\ell}$ that we want to compute are aggregate queries over a sliding window of length $\ell+1$ over the sequence $[cost(s_k, \ell+1)]_{1 \leq k \leq n}$. Depending on the utilitarian/egalitarian goal, the aggregate can be either summation or maximum, but in either case, all the $n - \ell$ aggregates can be computed in linear time using standard sliding window techniques. ◀

We are now ready to present our algorithm. We construct a weighted directed acyclic graph $\mathcal{G}$ corresponding to the problem instance, as follows. We are given a source node $s = (0,0)$ and a target node $t = (n+1, *)$. For the remaining vertices, we have one vertex for each pair (i,j) with $1 \leq i \leq n+1$ and $0 \leq j \leq \alpha$. Intuitively, vertex (i,j) has the meaning "agent $1 \leq i \leq n+1$ is the next one to consider[8] and so far we have excluded $0 \leq j \leq \alpha$ agents from the hike. The source s is connected with a directed edge towards $(1,0)$ and such an edge has weight 0, while t is reachable from the node $(n+1, j)$, for each $0 \leq j \leq \alpha$, via an edge of weight 0. For the remaining edges, we add the following two types:

- We add a weighted edge $(i,j) \xrightarrow{0} (i+1, j+1)$ for all $1 \leq i \leq n$ and $0 \leq j < \alpha$. Intuitively, these correspond to excluding agent i from the hike.
- We add a weighted edge $(i,j) \xrightarrow{c(i,k)} (k+1, j)$ for all $1 \leq i \leq k \leq n$ and $0 \leq j \leq \alpha$,. Intuitively, these correspond to adding a new coalition $C = \{i, i+1, \ldots, k\}$ to the hike, incurring a cost of $c(i,k)$.

The next lemma establishes how paths in $\mathcal{G}$ correspond to compact solutions to our problem and its statement immediately follows by the construction of $\mathcal{G}$.

[7] However, size-monotonicity is crucial in showing that considering compact solutions is enough.
[8] Indeed, there is a "dummy" agent $n+1$ signifying that there are no more agents to consider.

▶ **Lemma 16.** *There is a bijection from compact solutions to s-t paths in $\mathcal{G}$. Moreover, the social cost of a compact solution is the cost of the associated path, defined as either the sum or the maximum of the costs of its constituent edges.*

As a result, computing an optimal compact solution amounts to finding a minimum cost s-t path in $\mathcal{G}$; this gives us a polynomial time algorithm for computing an optimal solution.

▶ **Theorem 17.** *A hike with minimum social cost can be computed in time $O(n^2(\alpha + 1))$.*

Proof. By Lemma 14 it is enough to compute the best hike among compact solutions. To do so, we construct the graph $\mathcal{G}$ corresponding to the problem instance. Subsequently, we compute an s-t path in $\mathcal{G}$ of minimum cost. This can be done in time linear in the size of the graph, as the graph is acyclic. Correctness is assured by Lemma 16. For the time bound, note that the number of vertices in the graph is $O(n(\alpha + 1))$ and the number of edges is $O(n^2(\alpha + 1))$. Moreover, edge costs can be computed in constant time after $O(n^2)$ total precomputation by Lemma 15. Overall, we get a time complexity of $O(n^2(\alpha + 1))$. ◀

4 Wonderful Partitions Versus Minimum Egalitarian Partitions

So far we assumed each agent has an ideal group size, a peak, and there exists a cost function $cost(x, y)$ which expresses the cost that any agent having ideal group size x incurs if placed in a coalition of size y. More broadly, agents may express their disutility with any cost function, that is, for each agent i there is a mapping $cost_i : N \to \mathbb{R}$ simply expressing the cost agent i incurs when assigned to a coalition of a certain size.

In this section, we are interested in finding the minimum egalitarian cost achievable by any partition and we denote by Min-Eg the problem of computing this value. In Section 3, we have already shown that whenever $cost_i(s) = cost(s_i, s)$, Min-Eg can be computed in $O(n^2)$. In what follows, we exploit the connection between Wonderful-Partition and Min-Eg delineating the tractability of the latter with respect to the properties of the cost functions. First, we show a general connection between Wonderful-Partition and Min-Eg.

▶ **Proposition 18.** *Wonderful-Partition and Min-Eg are polynomial time equivalent.*

Proof. A Wonderful-Partition instance can be transformed into a Min-Eg instance by setting for each agent a cost function $cost_i$ where $cost_i(j) = 1$ if agent i does not approve coalition size j and $cost_i(j) = 0$, otherwise. Hence, the Wonderful-Partition instance is a yes instance if Min-Eg is 0 and it is a no instance, otherwise.

If we have a Min-Eg instance, there are at most n^2 distinct values $cost_i(j)$. Assume there are k distinct such values and let us sort them from the lowest to the highest, namely, $c_1 < \ldots < c_k$. For each value c_h, we can define a Wonderful-Partition instance where a coalition of size $j \in [n]$ is approved by agent i if and only if $cost_i(j) \leq c_h$. We can therefore determine if there exists a partition having egalitarian welfare of at most c_h by solving Wonderful-Partition on the just described instance. Clearly, if there exists a partition having an egalitarian cost of at most c then there exists a partition having an egalitarian cost of value at most c' for each $c \leq c'$. Conversely, if there is no partition having an egalitarian welfare of at most c then there is no partition having an egalitarian cost of at most c'', for each $0 \leq c'' \leq c$. Therefore, we can use binary search among the possible values $c_1, \ldots, c_k$ to find the minimum value c such that a partition having an egalitarian welfare of at most c exists. This solves Min-Eg. ◀

▶ **Corollary 19.** *If Wonderful-Partition can be decided in time $T(n)$, then Min-Eg can be solved in time $O((n^2 + T(n)) \cdot \log_2 n)$.*

Given the polynomial time equivalence between these two problems, it follows that solving MIN-EG is in general computationally intractable because WONDERFUL-PARTITION is NP-hard as soon as the approval sets are not intervals [17].

Nevertheless, there exists a special class of costs that can be solved using our dynamic programming approach for WONDERFUL-PARTITION described in Section 2.1. Such a class is a generalization of what we discussed in Section 3: Namely, each agent has an ideal group size s_i, and the closer the coalition size is to s_i the lower the cost. We align to the Hedonic Games literature calling this property *naturally single-peakedness.*[9]

▶ **Definition 20.** *A cost function* $cost : \mathbb{N} \to \mathbb{R}_{\geq 0}$ *is said to be* naturally single-peaked *if there exists an ideal group size, a* peak, $p \in [n]$ *such that* $h < k \leq p$ *or* $h > k \geq p$ *imply that* $cost(k) < cost(h)$ *holds.*

We observe that the reduction from MIN-EG to WONDERFUL-PARTITION described in Proposition 18 produces an interval instance in the case of naturally single-peaked cost functions. With this, we obtain the following theorem.

▶ **Theorem 21.** *MIN-EG for naturally single-peaked costs can be solved in time* $O(n^5 \log_2 n)$.

Proof. We can use a similar idea as in the polynomial time reduction from MIN-EG to WONDERFUL-PARTITION, as described in Proposition 18. In MIN-EG we are looking for a partition that minimizes the cost of the agent having the highest disutility. Here, since we consider naturally single-peaked preferences, the disutility of an agent i is decreased the closer the size of their assigned group $|\pi(i)|$ is to their ideal group size s_i (the peak value of agent i). We can now fix, for a value c, some distances Δ, Δ' such that $cost_i(s) \leq c$ for each $s \in [s_i - \Delta, s_i + \Delta']$. Therefore, we define that any agent i approves their assigned group size if it is in the interval $[s_i - \Delta, s_i + \Delta']$; this defines the instance of WONDERFUL-PARTITION-INTERVALS to be solved for determining if there exists a partition of having an egalitarian cost of at most c.

Now, applying Corollary 19 and Theorem 3, the statement follows. ◀

Notice that the computational complexity guaranteed by Theorem 17 is way more efficient than the one of Theorem 21. However, the former holds true for very specific naturally single-peaked costs, while the latter establishes the tractability of MIN-EG whenever agents have naturally single-peaked costs.

5 Conclusions

We resolved an open problem posed a decade ago by Gerhard Woeginger by giving a polynomial-time algorithm via establishing a connection to a version of rectangle stabbing, and investigated several interesting variants. We give a complete picture on the tractability of the Hiking Problem itself and show that maximizing the number of satisfied participants or deleting the minimal number such that the remaining participants admit a wonderful partition is polynomial time solvable. The tractability of both the original decision-, and the according optimization problems is crucially enabled by the existence of optimal solutions that exhibit simple and intuitive structural properties, fueling the algorithmic solutions based on Dynamic Programming. Last but not least, we employ our solution to efficiently compute

[9] We say *naturally* as general single-peakedness may be defined w.r.t. any fixed ordering of coalition sizes. In our setting, we consider cost functions that are single-peaked w.r.t. the natural ordering $1, \ldots, n$.

a partition that maximizes the egalitarian welfare for anonymous naturally single-peaked Hedonic Games. We note that our approach also works for general interval instances, that is, for a given permutation σ of numbers $1, \ldots, n$, intervals are defined over the numbers in order of the permutation, i.e., $\sigma(1), \ldots, \sigma(n)$. This extends our results from naturally single-peaked to general single-peaked cost functions. The problem of minimizing utilitarian cost for general single-peaked cost functions remains open.

References

1 José Alcalde and Pablo Revilla. Researching with whom? Stability and manipulation. *Journal of Mathematical Economics*, 40(8):869–887, 2004. doi:10.1016/j.jmateco.2003.12.001.

2 Haris Aziz, Florian Brandl, Felix Brandt, Paul Harrenstein, Martin Olsen, and Dominik Peters. Fractional hedonic games. *ACM Transactions on Economics and Computation*, 7(2):1–29, 2019. doi:10.1145/3327970.

3 Haris Aziz, Felix Brandt, and Paul Harrenstein. Pareto optimality in coalition formation. *Games and Economic Behavior*, 82:562–581, 2013. doi:10.1016/j.geb.2013.08.006.

4 Haris Aziz, Felix Brandt, and Hans Georg Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intellgence*, 195:316–334, 2013. doi:10.1016/J.ARTINT.2012.09.006.

5 Haris Aziz, Serge Gaspers, Joachim Gudmundsson, Julián Mestre, and Hanjo Täubig. Welfare maximization in fractional hedonic games. In *International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 461–467. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/071.

6 Haris Aziz, Paul Harrenstein, Jérôme Lang, and Michael J. Wooldridge. Boolean hedonic games. In *Principles of Knowledge Representation and Reasoning, KR 2016*, pages 166–175. AAAI Press, 2016. URL: http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12869.

7 Haris Aziz and Raul Savani. Hedonic games. In *Handbook of Computational Social Choice.* Handbook of Computational Social Choice. Cambridge University Press, 2016.

8 Coralio Ballester. Np-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, 2004. doi:10.1016/j.geb.2003.10.003.

9 Suryapratim Banerjee, Hideo Konishi, and Tayfun Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18:135–153, 2001. doi:10.1007/s003550000067.

10 Nadja Betzler, Arkadii Slinko, and Johannes Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013. doi:10.1613/JAIR.3896.

11 Davide Bilò, Vittorio Bilò, Pascal Lenzner, and Louise Molitor. Tolerance is necessary for stability: Single-peaked swap schelling games. In *International Joint Conference on Artificial Intelligence, IJCAI 2022*, pages 81–87, 2022. doi:10.24963/IJCAI.2022/12.

12 Duncan Black. On the rationale of group decision-making. *Journal of political economy*, 56(1):23–34, 1948. doi:10.1086/256633.

13 Niclas Boehmer and Edith Elkind. Individual-based stability in hedonic diversity games. In *AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 1822–1829, 2020. doi:10.1609/AAAI.V34I02.5549.

14 Anna Bogomolnaia and Matthew O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002. doi:10.1006/GAME.2001.0877.

15 Felix Brandt, Markus Brill, Edith Hemaspaandra, and Lane A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *Journal of Artificial Intelligence Research*, 53:439–496, 2015. doi:10.1613/JAIR.4647.

16 Robert Bredereck, Edith Elkind, and Ayumi Igarashi. Hedonic diversity games. In *International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019*, pages 565–573, 2019. URL: http://dl.acm.org/citation.cfm?id=3331741.

17 Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard Woeginger. Group activity selection problem with approval preferences. *International Journal of Game Theory*, 47(3):767–796, 2018. doi:10.1007/S00182-017-0596-4.

18 Jacques H Dreze and Joseph Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica: Journal of the Econometric Society*, pages 987–1003, 1980. doi:10.2307/1912943.

19 Edith Elkind, Piotr Faliszewski, and Piotr Skowron. A characterization of the single-peaked single-crossing domain. *Social Choice and Welfare*, 54(1):167–181, 2020. doi:10.1007/S00355-019-01216-3.

20 Edith Elkind and Michael J. Wooldridge. Hedonic coalition nets. In *International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS 2009*, pages 417–424, 2009. URL: https://dl.acm.org/citation.cfm?id=1558070.

21 Guy Even, Retsef Levi, Dror Rawitz, Baruch Schieber, Shimon Shahar, and Maxim Sviridenko. Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Transactions on Algorithms*, 4(3):34:1–34:17, 2008. doi:10.1145/1367064.1367074.

22 Tobias Friedrich, Pascal Lenzner, Louise Molitor, and Lars Seifert. Single-peaked jump schelling games. In *International Symposium on Algorithmic Game Theory, SAGT 2023*, pages 111–126, 2023. doi:10.1007/978-3-031-43254-5_7.

23 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* 1979.

24 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

25 Martin Hoefer, Sigal Oren, Roger Wattenhofer, and Giovanna Varricchio. Computational Social Dynamics (Dagstuhl Seminar 22452). *Dagstuhl Reports*, 12(11):28–44, 2023. URL: https://drops.dagstuhl.de/opus/volltexte/2023/17834, doi:10.4230/DagRep.12.11.28.

26 Jan Karel Lenstra, Franz Rendl, Frits Spieksma, and Marc Uetz. In memoriam Gerhard Woeginger. *Journal of Scheduling*, 25(5):503–505, 2022. doi:10.1007/s10951-022-00748-4.

27 Martin Olsen. Nash stability in additively separable hedonic games and community structures. *Theory of Computing Systems*, 45:917–925, 2009. doi:10.1007/s00224-009-9176-8.

28 Dominik Peters. Complexity of hedonic games with dichotomous preferences. In *AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 579–585. AAAI Press, 2016. doi:10.1609/AAAI.V30I1.10047.

29 Renate Schmid, 2011. © by Mathematisches Forschungsinstitut Oberwolfach gGmbH [Online; accessed November 01, 2023]. Published under CC-BY-SA 2.0. URL: https://opc.mfo.de/detail?photo_id=14972.

30 Toby Walsh. Uncertainty in preference elicitation and aggregation. In *AAAI Conference on Artificial Intelligence, AAAI 2007*, pages 3–8, 2007. URL: http://www.aaai.org/Library/AAAI/2007/aaai07-001.php.

31 Gerhard Woeginger. Core stability in hedonic coalition formation. In *International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2013*, pages 33–50. Springer, 2013. doi:10.1007/978-3-642-35843-2_4.

32 Lan Yu, Hau Chan, and Edith Elkind. Multiwinner elections under preferences that are single-peaked on a tree. In *International Joint Conference on Artificial Intelligence, IJCAI 2013*, pages 425–431, 2013. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6777.

A Proofs Omitted From Section 3

▶ **Lemma 12.** *The following properties hold:*

(a) *In the utilitarian setting, if $cost(\cdot,\cdot)$ is monotone and fulfills quadrangle inequality and reverse quadrangle inequality, then there exists a size-monotonic optimal solution.*

(b) *In the egalitarian setting, if $cost(\cdot,\cdot)$ is monotone, then there exists a size-monotonic optimal solution.*

Proof. We begin with property (a), i.e. the utilitarian case. Assume towards a contradiction that no optimal solution is size-monotonic. Let π be an optimal solution minimizing the number of pairs (i, j) of agents in π such that $i < j$ and $|\pi(i)| > |\pi(j)|$. We call such pairs *bad*. Since π is not size-monotonic, at least one bad pair exists and let (i, j) be such a bad pair. For ease of notation, denote $|\pi(i)|$ and $|\pi(j)|$ by a and b, respectively. Consider the alternative solution π' which is identical to π except that i and j swap coalitions. Note that π' has strictly fewer bad pairs than π. Now, consider

$$cost(\pi) - cost(\pi') = cost(s_i, a) + cost(s_j, b) - cost(s_i, b) - cost(s_j, a).$$

If we could show that this quantity is non-negative, then we would get that π' is also an optimal solution, but one having strictly less bad pairs, contradicting minimality. We will now show exactly this. Since $s_i \le s_j$ and $b < a$, it follows that we only need to consider the following six cases.

Case 1a: $s_i \le s_j \le b < a$**.**

The inequality $cost(s_i, a) + cost(s_j, b) \ge cost(s_i, b) + cost(s_j, a)$ follows immediately from the quadrangle inequality.

Case 2a: $s_i \le b \le s_j \le a$**.**

From quadrangle inequality we get $cost(s_i, a) + cost(b, b) \ge cost(s_i, b) + cost(b, a)$, i.e. the triangle inequality. By monotonicity we conclude

$$\begin{aligned} cost(s_i, a) + cost(s_j, b) &\ge cost(s_i, a) + cost(b, b) \ge cost(s_i, b) + cost(b, a) \\ &\ge cost(s_i, b) + cost(s_j, a). \end{aligned}$$

Case 3a: $s_i \le b < a \le s_j$**.**

By monotonicity we have $cost(s_i, a) \ge cost(s_i, b)$ and $cost(s_j, b) \ge cost(s_j, a)$. The inequality follows immediately.

Case 4a: $b \le s_i \le s_j \le a$**.**

We use again monotonicity to observe $cost(s_i, a) \ge cost(s_j, a)$ and $cost(s_j, b) \ge cost(s_i, b)$. This obtains the desired result.

Case 5a: $b \le s_i \le a < s_j$**.**

We use monotonicity and reverse quadrangle inequality to get

$$\begin{aligned} cost(s_i, a) + cost(s_j, b) &\ge cost(a, a) + cost(s_j, b) \ge cost(s_j, a) + cost(a, b) \\ &\ge cost(s_j, a) + cost(s_i, b). \end{aligned}$$

Case 6a: $b < a \le s_i \le s_j$**.**

This case follows directly from the definition of the reverse quadrangle property. This finishes the proof for the utilitarian case.

Now, to tackle property (b), the egalitarian case, the argument remains similar, except that now we need to show that $\max\{cost(s_i, a), cost(s_j, b)\} \ge \max\{cost(s_j, a), cost(s_i, b)\}$. We again use the case distinction as above but we only need to assume monotonicity of *cost*.

Case 1b: $s_i \leq s_j \leq b < a$.

By monotonicity we have

$$\max\{cost(s_i, a), cost(s_j, b)\} = cost(s_i, a) \geq \max\{cost(s_i, b), cost(s_j, a)\}$$

Case 2b: $s_i \leq b \leq s_j \leq a$.

We observe

$$\max\{cost(s_i, a), cost(s_j, b)\} \geq cost(s_i, a) \geq \max\{cost(s_i, b), cost(s_j, a)\}.$$

Case 3b: $s_i \leq b < a \leq s_j$.

By monotonicity we have $cost(s_i, a) \geq cost(s_i, b)$ and $cost(s_j, b) \geq cost(s_j, a)$. The inequality follows immediately.

Case 4b: $b \leq s_i \leq s_j \leq a$.

We use again monotonicity to observe $cost(s_i, a) \geq cost(s_j, a)$ and $cost(s_j, b) \geq cost(s_i, b)$. This obtains the desired result.

Case 5b: $b \leq s_i \leq a < s_j$.

We get

$$\max\{cost(s_i, a), cost(s_j, b)\} \geq cost(s_j, b) \geq \max\{cost(s_j, a), cost(s_i, b)\}.$$

Case 6b: $b < a \leq s_i \leq s_j$.

The last case can be obtained by

$$\max\{cost(s_i, a), cost(s_j, b)\} = cost(s_j, b) \geq \max\{cost(s_j, a), cost(s_i, b)\}.$$

This concludes the proof both for the utilitarian as well as the egalitarian case. ◀

▶ **Lemma 14.** *There exists an optimal partition which is size-monotonic and compact.*

Proof. Let π^* be an optimal size-monotonic partition, which, by Lemma 12, must exist. If partition π^* happens to be compact, then we are done. So, we can assume that partition π^* is not compact. Since compactness is violated, there exists a coalition $C \in \pi^*$, such that $\min\{h \mid h \in C\} = i$ and $\max\{h \mid h \in C\} = j$, and there exists an agent $k \notin C$ such that $i < k < j$. For agent k there are only two options:

(i) agent k participates in the hike, i.e., we have $\pi^*(k) = C' \neq C$. By size-monotonicity it follows that $|\pi^*(i)| = |\pi^*(j)| = |\pi^*(k)|$;

(ii) agent k does not participate in the hike, i.e., $\pi^*(k) = \perp$.

We will now perform transformation steps to partition π^* to ensure compactness without sacrificing size-monotonicity or optimality. Step 1 will remove compactness violations of type (i) and Step 2 will deal with type-(ii)-violations.

Step 1: Partition π^* contains coalitions of certain sizes. Given a coalition size $q \in [n]$, we rearrange the coalitions of size q in π^* in the following way: Let $C_1, C_2, \ldots, C_\ell$ be the coalitions in π^* of size q, and let $A_q = \bigcup_{1 \leq i \leq \ell} C_i = \{i_1, i_2, \ldots, i_{q\cdot\ell}\}$ be the set of agents that are in a coalition of size q in partition π^*. Moreover, we assume that the agents in A_q are sorted in increasing order of their ideal coalition sizes, i.e., we assume that $s_{i_1} \leq s_{i_2} \leq \cdots \leq s_{i_{q\cdot\ell}}$ holds. We now create ℓ many new coalitions of size q by reassigning the agents in A_q as follows: the first q agents $i_1, \ldots, i_q$ are assigned to coalition C_1^*, the next q agents $i_{q+1}, \ldots, i_{2q}$ are assigned to coalition C_2^*, and so on. The last q agents $i_{q\cdot(\ell-1)+1}, \ldots, i_{q\cdot\ell}$ are then assigned to coalition C_ℓ^*. Then we modify partition π^* by replacing the coalitions $C_1, \ldots, C_\ell$ with the coalitions $C_1^*, \ldots, C_\ell^*$. Note that after this replacement, partition π^* is still size-monotonic, since the respective coalition size stays the same for every agent in A_q. We iterate this replacement procedure for each coalition size q. We end up with a modified partition π^* that is still size-monotonic. furthermore, by construction, since we reassigned agents that are in coalitions having the same sizes in increasing order of their ideal coalition sizes, we cannot have compactness violations of type (i) in the final partition π^* at the end of Step 1.

Step 2: We will remove type-(ii)-violations of compactness one by one. For keeping track of our progress, we consider the following measure. Given a coalition C, the *diameter of* C is defined as $\operatorname{diam}(C) = \max\{h \mid h \in C\} - \min\{h \mid h \in C\}$. Moreover, the *diameter of a partition* π is $\operatorname{diam}(\pi) = \sum_{C\in\pi} \operatorname{diam}(C)$.

Assume that there is a coalition $C \in \pi^*$ with $\min\{h \mid h \in C\} = i$ and $\max\{h \mid h \in C\} = j$, that there exists an agent $k \notin C$ such that $i < k < j$, and that we have $\pi^*(k) = \perp$. There are two cases, depending on agent k's ideal coalition size s_k. If $s_k \geq |C|$, then we will change partition π^* by swapping agents k and j, i.e., agent j will be excluded from the hike and agent k will be assigned to coalition C instead. If $s_k < |C|$, then we swap agents i and k. Since $s_i \leq s_k \leq s_j$, none of those swaps can increase the social cost of partition π^*, i.e., partition π^* stays optimal, and the partition π^* stays size-monotonic. Moreover, in both cases the diameter of partition π^* is strictly decreased. This implies that after finitely many such exchange steps, this process must stop and all type-(ii)-violations are resolved.

It remains to show that the exchanges done in Step 2 do not create new type-(i)-violations of compactness. We show this via a proof by contradiction. Assume that for some coalition $C \in \pi^*$ with $\min\{h \mid h \in C\} = i$ and $\max\{h \mid h \in C\} = j$ we have exchanged agent k with agent j in Step 2 and this creates a new type-(i)-violation of compactness, that is, there is some agent k' with $i < k' < j'$, where $j' = \max\{h \mid h \in C\}$ after the exchange. However, also before the exchange of j and k, we had that $i < k' < j' < j$ holds, which implies that agent k' already was a type-(i)-violation of compactness. This contradicts that a new type-(i)-violation was introduced. The argument for the case where agents i and k are exchanged is completely analogous.

Thus, by first performing Step 1 to remove all type-(i)-violations of compactness and then performing Step 2 to remove all type-(ii)-violations of compactness we will eventually transform the optimal size-monotonic partition π^* into an optimal partition that this still size-monotonic but also compact. ◀

B Hardness for Approval Sets of Size 2

We have shown that whenever the approval set of each agent is an interval, solving Wonderful-Partition is possible in polynomial time. Clearly, whenever all agents have approval sets of size 1, the existence of a wonderful partition is polynomial-time decidable as well: This can also be seen as a special case of interval instances (although it can also be solved directly, as explained in the introduction). In general, the approval sets are not necessarily intervals, and without any assumption on the structure of the approval sets, the Wonderful-Partition problem is NP-complete even if the size of each approval set is at most 3. This follows from the hardness proof of Woeginger [31] and the fact that

Exact Cover By 3-Sets is NP-hard (see below for the definition).[10] Darmann et al. [17] establish an even stronger version of this result for the case where all approval sets are of size at most 2.

In this section, we precisely map the boundary of tractability of Wonderful-Partition with respect to the approval set size. While the case with approval set size 1 can be solved in polynomial time, we now show NP-completeness if the approval sets have size *exactly* 2. In particular, we will show the following:

▶ **Theorem 22.** *Deciding Wonderful-Partition is NP-complete, even if the approval sets are of size* exactly *2.*

We first observe that Wonderful-Partition when restricted to the case of approval sets of size exactly 2, is equivalent to a graph-theoretic problem we call Orientation. For this, we write $d^+(v)$ to denote the in-degree of a vertex $v \in V$ in a directed graph $\vec{G} = (V, \vec{E})$.

Orientation
Input: An undirected graph $G = (V, E)$ with $V = [n]$, admitting parallel edges but no self-loops.
Question: Does there exist an orientation of the edges $\vec{G}$ such that $d^+(i) \equiv 0 \pmod{i}$ for each $i \in V$?

We now show that the two problems are equivalent.

Indeed, an instance of Wonderful-Partition where all agents approve exactly two sizes can be transformed into an equivalent instance of Orientation by representing each agent approving sizes $i, j \in [n]$ by an undirected edge (i, j). Orienting this edge to either node i or node j models that the respective agent is part of a partition of size i or j, respectively. If $d^+(i) = k \cdot i$ for some $k \geq 0$ then this means that k partitions with size i will be created. Conversely, an instance of Orientation with m edges can be transformed to an instance of Wonderful-Partition with m agents, where each edge (i, j) corresponds to an agent with approval set $\{i, j\}$.

Using the equivalence between Wonderful-Partition and Orientation, we will prove Theorem 22 by reducing Exact Cover By 3-Sets (X3C) to Orientation. X3C is well-known to be NP-hard [23] and it is defined as follows:

Exact Cover By 3-Sets (X3C)
Input: A ground set $X = \{x_1, \ldots, x_{3k}\}$ and a collection $\mathcal{C}$ of 3-element subsets (triples) of X.
Question: Does there exist a subset $\mathcal{C}' \subseteq \mathcal{C}$ such that $\bigcup_{C \in \mathcal{C}'} C = X$ and $|\mathcal{C}'| = k$, i.e., $\mathcal{C}'$ is an exact cover of X?

Proof of Theorem 22. Let us assume that $|\mathcal{C}| = q$ and let us enumerate all the triples as $C_1, \ldots, C_q$. For simplicity, we fix $p = 3k$. Notice that any element x in the ground set can be contained in at most $\binom{p-1}{2} < p^2$ triples. Let us denote by δ the highest number of such occurrences. Our reduction is as follows:

[10] In fact, as a curiosity, even if every element of the ground set appears in exactly three triples and any two triples overlap in at most one element [24].

- For each element in the ground set $x_i \in X$, for $i \in [p]$, we create an *element gadget* which consists of nodes $i \cdot \delta$ and $i \cdot \delta + 1$, respectively. Such two nodes are connected by $i \cdot \delta - 1$ many edges. We may refer to these edges as element-(x_i)-edges. Moreover, we call node $i \cdot \delta$ the element-(x_i)-node.
- For each triple $C_j \in \mathcal{C}$, for $j \in [q]$, we create a *triple gadget* which consists of two nodes having value $p \cdot \delta + 4j$ and $p \cdot \delta + 4j - 3$, respectively. Such two nodes are connected by $p \cdot \delta + 4j - 3$ many edges. We will refer to these edges as triple-(C_j)-edges. Moreover, we call node $p \cdot \delta + 4j$ the triple-(C_j)-node.
- Finally, for each $x_i \in X$ and $C_j \in \mathcal{C}$ such that $x_i \in C_j$, we connect the nodes $i \cdot \delta$ and $p \cdot \delta + 4j$ with one edge.

Clearly, the reduction is polynomial. Moreover, the constructed graph is well-defined (there is no overlap in gadgets corresponding to different compounds). See Figure 3 for an example.

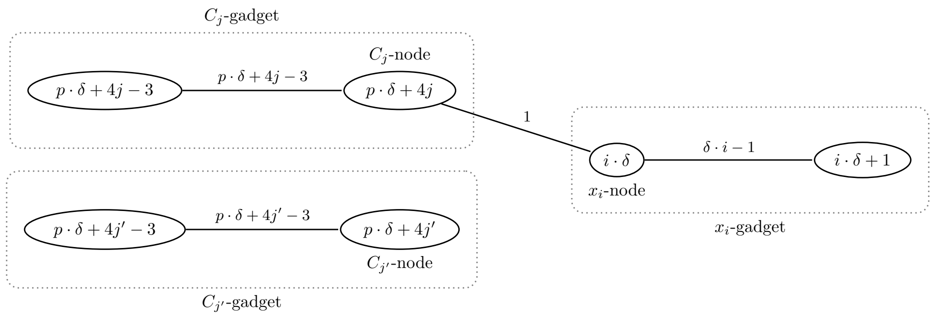

Figure 3 Gadgets – On the left, above (resp. below) the triple gadget for C_j (resp. $C_{j'}$); on the right, the element gadget for x_i. The picture shows the set-up of the gadgets if $x_i \in C_j$ but $x_i \notin C_{j'}$. Labels on edges represent the number of parallel edges between the two nodes.

The idea is to determine the exact cover by means of the orientation of edges connecting elements and triple gadgets. Specifically, whenever a triple C_j is in the covering set, the three edges connecting the gadget C_j with the corresponding element gadgets must all be oriented towards the corresponding element nodes. In turn, if C_j is not in the covering set, the orientation of these edges must be towards the triple-(C_j)-node.

With this, it follows that whenever an exact cover exists, an orientation of G exists as well. In particular, for any triple C_j in the covering set we can orient all edges incident to the triple-(C_j)-node away from the triple-(C_j)-node, and in the opposite direction, otherwise. Moreover, every element edge is directed towards the corresponding element node.

It remains to show that whenever an orientation exists, an exact cover exists as well. The rest of this proof is established by the following observations:

1. In any feasible orientation, the element edges are always oriented toward the element node. Otherwise, the orientation will not be feasible since the non-element node endpoint of every element edge has a label that is higher than the number of its incident edges.
2. Denote by t_i the number of triples containing x_i, the element-(x_i)-node has $i \cdot \delta - 1 + t_i$ incident edges. By (1), the in-degree in a feasible orientation is at least $i \cdot \delta - 1$. Since $t_i \leq \delta$, in a feasible orientation of G the in-degree of $i \cdot \delta$ is exactly its value. As a consequence, there is only one incoming edge from triple gadgets, and therefore each element is covered by exactly one triple.
3. Last but not least, in a feasible orientation, a triple cannot be only partially used. Specifically, for a triple C_j with triple-(C_j)-node v either all edges incident to v must

be all oriented to v or all away from v. This is ensured by the fact that in any feasible orientation the in-degree of a triple C_j is either 0 or $p \cdot \delta + 4j$. ◀