symmetry

MDPI

Article

Breakout Group Allocation Schedules and the Social Golfer Problem with Adjacent Group Sizes

Alice Miller *, Matthew Barr, William Kavanagh, Ivaylo Valkov and Helen C. Purchase

School of Computing Science, University of Glasgow, Glasgow G12 8RZ, UK; matthew.barr@glasgow.ac.uk (M.B.); w.kavanagh.1@research.gla.ac.uk (W.K.); ivaylo.valkov@glasgow.ac.uk (I.V.); helen.purchase@glasgow.ac.uk (H.C.P.)
* Correspondence: alice.miller@glasgow.ac.uk

Abstract: The current pandemic has led schools and universities to turn to online meeting software solutions such as Zoom and Microsoft Teams. The teaching experience can be enhanced via the use of breakout rooms for small group interaction. Over the course of a class (or over several classes), the class will be allocated to breakout groups multiple times over several rounds. It is desirable to mix the groups as much as possible, the ideal being that no two students appear in the same group in more than one round. In this paper, we discuss how the problem of scheduling balanced allocations of students to sequential breakout rooms directly corresponds to a novel variation of a well-known problem in combinatorics (the social golfer problem), which we call the social golfer problem with adjacent group sizes. We explain how solutions to this problem can be obtained using constructions from combinatorial design theory and how they can be used to obtain good, balanced breakout room allocation schedules. We present our solutions for up to 50 students and introduce an online resource that educators can access to immediately generate suitable allocation schedules.

Keywords: online meeting software; breakout room; small group teaching; block design; scheduling; combinatorics; social golfer problem

Citation: Miller, A.; Barr, M.; Kavanagh, B.; Valkov, I.; Purchase, H.C. Breakout Group Allocation Schedules and the Social Golfer Problem with Adjacent Group Sizes. *Symmetry* **2021**, *13*, 13. https://dx.doi.org/10.3390/sym13010013

Received: 3 December 2020
Accepted: 15 December 2020
Published: 23 December 2020

1. Introduction

Group work has long been a staple of computing science (CS) education, not least because it reflects how software is developed in industry [1]. The pedagogical advantages of group work are also well documented. Collaborative learning, whereby students work in groups together towards a specified goal [2], has been shown to help students to develop critical thinking skills [3] and a shared understanding of the studied material [4], and to enhance students' interpersonal, social, and teamwork skills [5]. The increasingly ubiquitous flipped classroom approach [6] requires that students access learning materials in advance of the class, so that the in-class time is spent on "interactive group-based learning activities" [7]. With an essential requirement of group work, it has been shown to improve learning performance [8], enhance students' enjoyment [9], and to be effective for large classes [10]. Furthermore, from a constructivist viewpoint, providing opportunities for social interaction via group work is an important component of CS education [11]. Peer instruction, wherein students discuss problems with their peers to construct their conceptual understanding, is also understood to be an effective instructional approach in CS education [12,13]. Perhaps most often associated with introductory programming tuition [14–16], peer instruction has also been shown to be an effective pedagogical technique with more advanced CS cohorts [17].

Using online meeting software such as Zoom and Microsoft Teams to deliver teaching has become the norm during the COVID-19 pandemic, with breakout rooms used to facilitate small group activities, especially where class sizes are large. Where class sizes are small enough, and the composition of the cohort is well understood in terms of students' abilities, experience, and personalities, groups may be allocated manually. For example,

Ref. [18] describes how manually constructed breakout groups comprising five or six students of mixed ability were used on a software engineering programme during the COVID-19 pandemic. The carefully constructed composition of the groups in this case could be fixed, with students remaining in the same groups for each round of group activities. However, such an approach is not practicable with the larger cohorts that are typical of CS university programmes. Furthermore, the repeated manual allocation of students to new groups ensuring the desired properties is prohibitively time-consuming, and not practical to administer in class, in real time.

Where manual allocation is not possible or practical, the facility to randomise allocation—as offered by tools such as Zoom—is a tempting alternative. Random selection works well once, for the first round, but subsequent random allocations are likely to allocate participants to groups with people they have been in a group with in a previous round. This is exacerbated as the number of rounds increases. Such an approach is not ideal for at least two reasons: first, problems associated with personality clashes between students will be amplified by repeated allocation to the same groups; second, any combinations where the difference in students' abilities proves problematic will persist from one round to the next.

Thus, the ideal solution is to programmatically assign students to new groups for each round, such that they are not allocated to groups with the same fellow students and, ideally, no two students appear in the same group in more than one round. Furthermore, prior literature suggests that having groups of four to six students is optimal [19–21], so this is assumed here.

Zoom currently has two options for allocating participants to breakout sessions, either random or set by the host (or co-host). If the latter option is chosen, individual rounds can be imported in the form of CSV files consisting of lists of participants' email addresses which correspond to their Zoom login IDs. Microsoft Teams also features breakout rooms (and has done since October 2020) and permits automated random allocation or manual assignment.

If random breakout rooms are used, it is difficult to maintain balance between rounds. Consider, for example, a class of 16 students. The host wants to allocate the students to 4 breakout rooms of size 4, and do this for 5 rounds. If we label the participants by numbers $0, 1, \ldots, 15$ and allocate the groups randomly, an example of the successive rounds generated is as follows:

round 1:	$(1,2,5,10)$,	$(6,7,9,12)$,	$(3,4,11,15)$,	$(0,8,13,14)$
round 2:	$(1,5,8,12)$,	$(2,4,6,10)$,	$(0,7,14,15)$,	$(3,7,9,13)$
round 3:	$(2,3,4,6)$,	$(8,11,13,14)$,	$(1,7,9,10)$,	$(0,5,12,15)$
round 4:	$(5,6,9,13)$,	$(3,4,7,8)$,	$(2,10,11,12)$,	$(0,1,14,15)$
round 5:	$(4,5,6,12)$,	$(9,11,13,14)$,	$(0,2,8,15)$,	$(1,3,7,10)$

Note that, for example, students 0 and 15 are allocated to the same group 4 times (in rounds 2, 3, 4 and 5). If the groups are allocated in a balanced way, this pair should be allocated to the same group at most once.

Neither Zoom or Microsoft Teams has functionality for allocation over multiple rounds, let alone for balanced allocation. However, scheduling the rounds in such a way as to be balanced by hand is extremely difficult, even for a relatively small number of students like 16.

Allocating items to groups in ways which have defined properties has been a subject of mathematical study for many years, namely combinatorial design theory. Designs (or *block designs* [22]—see Section 4.1) are mathematical structures in which points are allocated to blocks. This problem can be directly mapped onto our breakout room allocation problem (where points are participants and blocks are groups).

To solve the example above, we need a particular type of mathematical structure called a Kirkman system. Fortunately for us, such a structure exists for 16 points and gives a solution to our round allocation as follows:

round 1:	$(0,1,2,3)$,	$(4,5,6,7)$,	$(8,9,10,11)$,	$(12,13,14,15)$
round 2:	$(0,7,10,13)$,	$(3,4,9,14)$,	$(2,5,8,15)$,	$(1,6,11,12)$
round 3:	$(0,5,11,14)$,	$(1,4,10,15)$,	$(3,6,8,13)$,	$(2,7,9,12)$
round 4:	$(0,6,9,15)$,	$(2,4,11,13)$,	$(1,7,8,14)$,	$(3,5,10,12)$
round 5:	$(0,4,8,12)$,	$(1,5,9,13)$,	$(2,6,10,14)$,	$(3,7,11,15)$

In this case, each participant is in a group with every other participant exactly once.

Our example is chosen to both illustrate the difficulty of scheduling sequential allocations and the existence of an ideal solution. If fewer than 5 rounds were required, the same solution could be used, but some rounds ignored.

What if, for the number of participants, there is no suitable Kirkman System? Indeed, what if the number of participants is prime, so it is not possible to allocate the same number of participants to each block, even for a single round? In this paper, we consider what makes a good, balanced allocation schedule for the given parameters (number of participants, maximum number of rounds) and introduce a new combinatorial problem, the social golfer problem with adjacent group sizes (SGA). With additional constraints, solutions to SGA provide suitable good, balanced breakout room allocation schedules.

Social golfer-type problems are inherently linked to symmetry. Combinatorial search for solutions is hard due to the number of equivalent partial solutions at each level of search. Symmetry breaking [23–28] must be used to eliminate these equivalent structures and reduce the search space to a tractable size. In addition, although we do not describe them in this paper, the constructions we rely on in Section 4 to generate Kirkman systems, nearly Kirkman systems and resolvable transversal designs, involve finite (Galois) groups to create parallel classes of blocks from a set of base blocks. The nature of finite groups prevents the repetition of pairs in different blocks by avoiding undesired periodicity due to symmetry.

Using existing constructions from design theory, we show how solutions to SGA can be obtained for up to 50 participants. We introduce an online tool which allows educators to generate good, balanced allocation schedules—currently for up to 70 participants.

2. Good, Balanced Allocation Schedules

In this paper, we want to develop breakout room allocation schedules, that is, predefined sequences of rounds of participants allocated to groups. We call such an allocation schedule balanced when no two participants are allocated to a group together more than once.

The natural way to divide a set of participants into groups is to use equal group sizes. However, this is not always possible (if the number of participants is prime, say). However it makes sense to limit the number of group sizes to as small a number as possible, and to keep the difference in group sizes to a minimum. We therefore allow at most two group sizes and insist that the group sizes differ by at most one.

As previously mentioned, groups of four to six students are optimal [19–21]. However, in some cases, this may be either impossible (when the number of participants is too small) or overly restrictive (when the number of participants is perfectly divisible by a number p greater than 2, but outside of this range, it might be preferable to have equal groups of size p rather than unequal group sizes within the desired range). We therefore define a good, balanced allocation schedule as follows:

Definition 1. *For a set of v participants, a good balanced allocation schedule is a balanced allocation schedule for which:*

1. *there are at most* 2 *group sizes,*

2. *group sizes differ by at most* 1,
3. *every round has the same distribution of group sizes,*
4. *if* $v \geq 12$ *all groups have a size of at least* 3
5. *If* $v \geq 20$ *and there are two group sizes, then all groups have sizes between* 4 *and* 6.

3. The Social Golfer Problem and a Variation

The problem of allocating a set of v participants to equal groups of size k in r rounds in such a way that every participant appears in every round and all pairs of participants appear in at most 1 group is a popular problem known as the social golfer problem (which we will refer to as $SGP(v, k, r)$) [29,30]. It originated from the following question posed in 1998 to `sci.op-research`:

> 32 golfers play golf once a week, and always in groups of 4. For how many weeks can they play such that no two players play together more than once in the same group?

This problem and its generalisation to any number of players v and group size k have attracted particular attention in the combinatorial optimisation community, due to its highly combinatorial and symmetric nature. It has become a standard benchmark example to evaluate modelling and symmetry breaking techniques [25,31]. Techniques for solving the social golfer problem include heuristic approaches [32], Boolean satisfiability (SAT) encoding [33,34] and constraint-based techniques [23,35,36]. We do not provide details of these approaches here, but excellent surveys can be found in [29], and more recently in [34,37].

The link between the social golfer problem and combinatorial designs, specifically mutually orthogonal Latin squares [22,38], Kirkman systems [22,39,40] and resolvable group divisible designs [22], is well known [29,31,41]. In this paper, we will refer to published best-known solutions to the social golfer problem [42,43], both for allocation schedules where there is only one group size, and to construct allocation schedules with two different group sizes, by adding or removing points to/from existing social golfer solutions.

Generally, the social golfer problem refers to fixed values of v and k and finding the largest r for which $SGP(v, k, r)$ has a solution. In our context of allocation schedules, we may not require a maximal solution (as often only a few rounds are required, or the maximal solution would provide too many rounds to feasibly implement). However, finding the maximal solution (or the largest known solution) will allow any number of rounds r' up to this value to be obtained by choosing only the first r' rounds:

Lemma 1. *If there is a solution for* $SG(v, k, r)$*, then there is a solution for* $SG(v, k, r')$*, for any* r' *less than* r*.*

The Social Golfer Problem with Adjacent Group Sizes

Our problem, though, is broader. We want to find a (possibly maximal) set of rounds in the same way, but we do not insist that all groups have the same size. Indeed, in many cases, this would not be possible (if the number of participants is prime, for example). We do insist however that, for v sufficiently large, all rounds have the same distribution of group sizes, and there are at most 2 group sizes, which differ by at most 1.

We call this problem the social golfer problem with adjacent group sizes, which we define as follows:

Definition 2. *Let* K *be a set of positive integers of size at most* 2 *and, if* $|K| = 2$*, then* $K = \{k, k+1\}$ *for some* k*. Then, for a given* v *and* r*, the social golfer problem with adjacent group sizes,* $SGA(v, K, r)$*, is to allocate* v *participants to groups with sizes from* K *in* r *rounds in such a way that every participant appears in every round, all pairs of participants appear in at most* 1 *group, and all rounds have the same set of group sizes.*

Note that, if $|K| = 1$, $SGA(n, K, r)$ is simply $SGP(v, k, r)$ for some k. As for the social golfer problem, we can construct solutions for any number of rounds r' less than the maximal number (or a maximum known number) of rounds, by choosing only the first r' rounds:

Lemma 2. *If there is a solution for $SGA(v, K, r)$, then there is a solution for $SGA(v, K, r')$, for any r' less than r.*

Note that for a given v, there can be many possibilities for K and thus a large number of ways to allocate participants to rounds. Our definition of a good balanced allocation (Definition 1) allows us to impose additional constraints on the group sizes and so reduce the available allocations. In the rest of this paper, we use SGP and SGA to refer to the classes of social golfer problems and social golfer problem with adjacent group sizes (i.e., when we do not wish to specify a particular set of parameters).

In Section 4.1, we introduce some techniques that we use to construct solutions for the SGA, which correspond to good balanced allocation schedules.

4. Finding Solutions to the SGA Using Results from Combinatorial Design Theory

The problem of finding solutions to the SGA is closely linked to the field of combinatorial block designs [22]. In this case, a set of points V are placed into blocks in such a way that no pair of points appear in more than one block together. In this section, we provide some preliminary definitions and examples from the field of block designs and describe how we can find solutions to SGA from existing block designs and solutions to the social golfer problem.

4.1. Preliminary Definitions And Examples

Our requirement to schedule groups of participants into a set of rounds in which every participant appears is equivalent to a parallel assignment of blocks in a design. Example 1 illustrates this equivalence.

Example 1. *Suppose we have a set of 9 participants who we want to arrange into 4 rounds, each containing groups of 3 students. In design theory terms, this is equivalent to saying that we have a set V of 9 points, which we will label $0, 1, \ldots, 8$, that we want to arrange into 4 parallel sets of 3 blocks, where no pair appears in more than one block. A solution to this problem is given below:*

round 1:	$(0,1,2)$,	$(3,6,7)$,	$(4,5,8)$
round 2:	$(0,3,4)$,	$(1,6,8)$,	$(2,5,7)$
round 3:	$(0,5,6)$,	$(1,4,7)$,	$(2,3,8)$
round 4:	$(0,7,8)$,	$(1,3,5)$,	$(2,4,6)$

In the rest of this section, we will refer to points and blocks rather than participants or students and groups, as we will be referring to structures within the context of block designs. Additionally (as can be seen below), the term group has a different meaning within this context.

In Example 1, every block has the same size, every point is in exactly one block with every other point, and the blocks can be arranged into rounds with every point appearing once per round. Designs with these properties are known as Kirkman systems [22] and, in particular, when the blocks all have size 3, as Kirkman triple systems [39,40]. This type of design will be useful for us, particularly because various constructions exist in the literature to construct them for certain parameters (number of points and size of blocks). However, they only exist for some parameters.

We refer to a Kirkman system on v points with blocks of size k as a $KS(v, k)$ and a Kirkman triple system on v points as a $KTS(v)$.

The following lemma shows the values of v and k, for $v \leq$ and $3 \leq k \leq 7$, for which a $KS(v, k)$ exists. These values can be calculated from results in [22] and references therein:

A closely related type of design of which we make use, is a resolvable group divisible design [44]. In this case, the blocks can again be placed into parallel classes, but not all possible pairs are contained in the blocks; instead, some of the pairs are contained within another set of non-intersecting subsets of V known as groups. If the groups all have the same size g (which we shall assume hereafter), we refer to a $RGDD(v,k,g)$ [45]. An $RGDD(v,3,2)$ is known as a nearly Kirkman triple system ($NKTS(v)$).

Example 2. *Suppose we want to create a solution to* $SGA(20,\{4\},r)$ *for* r *as large as possible. There is no* $KS(20,4)$*, so we cannot achieve a set of rounds in which every pair occurs exactly once. However, we can achieve* 5 *rounds of blocks, and the unused pairs can be arranged into a set of* 4 *groups of size* 5*. This is an example of an* $RGDD(20,4,5)$*:*

round 1:	$(0,1,2,3)$,	$(4,5,6,7)$,	$(8,9,10,11)$,
	$(12,13,14,15)$,	$(16,17,18,19)$	
round 2:	$(0,5,10,15)$,	$(4,9,14,19)$,	$(3,8,13,18)$,
	$(2,7,12,17)$,	$(1,6,11,16)$	
round 3:	$(0,7,9,18)$,	$(2,4,11,13)$,	$(6,8,15,17)$,
	$(1,10,12,19)$,	$(3,5,14,16)$	
round 4:	$(0,6,13,19)$,	$(3,4,10,17)$,	$(1,7,8,14)$,
	$(5,11,12,18)$,	$(2,9,15,16)$	
round 5:	$(0,11,14,17)$,	$(1,4,15,18)$,	$(2,5,8,19)$,
	$(3,6,9,12)$,	$(7,10,13,16)$	
groups:	$(0,4,8,12,16)$,	$(1,5,9,13,17)$,	$(2,6,10,14,18)$,
	$(3,7,11,15,19)$		

The blocks of this design give a $SGA(20,\{4\},5)$*. There is no* $SGA(20,\{4\},6)$ *(see Lemma 3).*

An $RGDD(v,k,1)$ is a $KS(v,k)$ as all pairs will appear in the blocks. Similarly, an $RGDD(v,k,k)$ can be viewed as a $KS(v,k)$ by treating the groups as additional blocks.

A particular type of resolvable group divisible design that we employ in this paper is one for which the number of groups is equal to the block size. In this case, every group contains a single point from every block (and vice versa) and the design is known as a resolvable transversal design. If the block size is k and the number of groups is n, we refer to an $RTD(k,n)$. The number of points in this case is $k*n$ and the number of parallel rounds of blocks is n. This type of design is very useful because they can be generated easily using an existing construction technique (if they exist for the given parameters) [22]. Notice that Example 2 is an $RTD(4,5)$.

4.2. Constructing Solutions for Sga

In many cases, when our parameters allow, we use existing Kirkman systems and resolvable group divisible designs and solutions to SGP to find solutions to SGA. In other cases, we add or remove points to/from blocks and groups of existing structures.

In order to produce our solutions, we need to access existing designs. We use the following resources:

1. Constructions for Kirkman triple systems [40],
2. Resolvable transversal design construction using mutually orthogonal Latin squares [22],
3. Online current best solutions to examples of the social golfer problem [43],
4. Published small Kirkman systems [46] and nearly Kirkman triple systems [47,48].

We do not go into detail here about the constructions and solutions listed above. We use the solutions and our own implementation of the constructions to create the designs on which our SGA solutions are built. Some useful results are given in Lemma 3.

Lemma 3. *The following results hold:*

1. *If a $KS(v,k)$ exists then k divides v and $k-1$ divides $v-1$.*
2. *A $KTS(v,k)$ exists if and only if $v \equiv 3 (\mathrm{mod}\ 6)$ and $v \geq 9$. An $NKTS(v)$ exists if and only if $v \equiv 0 (\mathrm{mod}\ 6)$ and $v \geq 18$ [40,49].*
3. *If there is a solution to $SGA(v,K,r)$ where $K = \{k, k+1\}$ and where, for every round, there are $m_1 > 0$ blocks of size k and $m_2 \geq 0$ blocks of size $k+1$, then r is at most $R(v,k,m_1,m_2))$, the largest integer less than or equal to $v(v-1)/(k(m1(k-1)+m2(k+1))$.*
4. *If $k = 4$ and $v = 20$ there is no solution to $SGP(20,4,r)$ with $r = R(20,4,5,0)$ [50].*
5. *There is no solution to $SGP(36,6,r)$ for $r > 3$ (a consequence of [51]).*

In the following example, we show how a solution to $SGA(13,\{3,4\},4)$ can be obtained by using an existing KS and removing points and blocks. We will use similar constructions for many of our SGA solutions. Note that we continue to use terms from block design theory throughout (points and blocks) to avoid confusion.

Example 3. *Suppose we want to create a solution to $SGA(13,\{3,4\},r)$. The only way to do this is for every round to have 3 blocks of size 3 and one of size 4. In order to construct such a set of blocks, we can start with a $KS(16,4)$ thus:*

round 1:	$(0,4,8,12)$,	$(1,5,9,\mathbf{13})$,	$(2,6,10,\mathbf{14})$,	$(3,7,11,\mathbf{15})$
round 2:	$(0,7,10,\mathbf{13})$,	$(1,6,11,12)$,	$(2,5,8,\mathbf{15})$,	$(3,4,9,\mathbf{14})$
round 3:	$(0,5,11,\mathbf{14})$,	$(1,4,10,\mathbf{15})$,	$(2,7,9,12)$,	$(3,6,8,\mathbf{13})$
round 4:	$(0,6,9,\mathbf{15})$,	$(1,7,8,\mathbf{14})$,	$(2,4,11,\mathbf{13})$,	$(3,5,10,12)$
round 5:	$(0,1,2,3)$,	$(4,5,6,7)$,	$(8,9,10,11)$,	$(12,\mathbf{13},\mathbf{14},\mathbf{15})$

In order to construct rounds on 13 points only, we remove 3 points from the last block (i.e., the points in bold). This destroys the last block, so we will disregard the last round of blocks in our allocation schedule. These 3 points will not appear together in any other block, so we can be assured that no other block will be reduced to size less than 3. Removing these points and the final round of blocks, we obtain the following solution to $SGA(13,\{3,4\},4)$:

round 1:	$(0,4,8,12)(1,5,9)(2,6,10)(3,7,11)$
round 2:	$(0,7,10)(1,6,11,12)(2,5,8)(3,4,9)$
round 3:	$(0,5,11)(1,4,10)(2,7,9,12)(3,6,8)$
round 4:	$(0,6,9)(1,7,8)(2,4,11)(3,5,10,12)$

We can use this technique to remove up to $k+1$ points from any existing $KS(v,k+1)$ and obtain solutions to $SG(v',\{k,k+1\},r)$ for $v-k-1 \leq v' \leq v-1$ and appropriate values of r. We always remove points from the last block in the final round of the $KS(v,k+1)$. If more than 1 point is to be removed, then the blocks from the final round of the $KS(v,k+1)$ should not be included in the solution. If the points removed do not happen to be the largest points (as they were in Example 3), then after every point is removed, the remaining points should be appropriately renumbered. It is also possible to remove more than $k+1$ points when the number of blocks per round is equal to $k+1$. We only see one example of this (the first case where $v = 11$ in Section 5.1), so we do not go into any detail about this possibility here. We can use a similar approach to obtain solutions to $SG(v',\{k,k+1\},r)$ for $v-n \leq v' \leq v-1$ and appropriate values of r from any $RTD(k+1,n)$. In this case, we can remove up to n points from the final group. As the groups are never used in our solutions (only the blocks), the maximum value of r in our solutions is the same as the number of rounds for the original $RTD(k+1,n)$ (i.e., n).

In some cases we can obtain solutions to SGA by adding points to existing structures.

Example 4. *Suppose we want to create a solution to* $SGA(19, \{3,4\}, 6)$. *One way to do this is for every round to have* 5 *blocks of size* 3 *and one of size* 4. *In order to construct a suitable set of blocks, we can start with an* $NKTS(18)$. *This is a set of parallel blocks of size* 3, *where every point is in a block with all but one of the other points:*

round 1: $(\mathbf{0},\mathbf{2},\mathbf{3})(1,15,17)(5,6,11)(4,7,13)(8,12,14)(9,10,16)$
round 2: $(\mathbf{1},\mathbf{4},\mathbf{16})(0,4,5)(3,8,10)(2,9,12)(6,13,15)(7,11,17)$
round 3: $(0,6,7)(1,11,13)(4,12,16)(5,10,14)(3,9,17)(2,8,15)$
round 4: $(1,10,12)(0,8,9)(2,13,17)(3,11,15)(5,7,16)(4,6,14)$
round 5: $(4,9,15)(2,7,14)(\mathbf{6},\mathbf{12},\mathbf{17})(8,13,16)(0,10,11)(1,3,5)$
round 6: $(5,8,17)(3,6,16)(\mathbf{7},\mathbf{10},\mathbf{15})(9,11,14)(1,2,4)(0,12,13)$
round 7: $(2,11,16)(4,10,17)(\mathbf{5},\mathbf{9},\mathbf{13})(3,7,12)(0,14,15)(1,6,8)$
round 8: $(3,13,14)(5,12,15)(\mathbf{4},\mathbf{8},\mathbf{11})(2,6,10)(1,7,9)(0,16,17)$

Observe that the blocks in boldface are all from different rounds and do not intersect. We can therefore add a new point to these blocks and, including only those rounds for which the new point has been added, obtain a solution to $SGA(19, \{3,4\}, 6)$

round 1: $(0,2,3,18)(1,15,17)(5,6,11)(4,7,13)(8,12,14)(9,10,16)$
round 2: $(1,4,16,18)(0,4,5)(3,8,10)(2,9,12)(6,13,15)(7,11,17)$
round 3: $(4,9,15)(2,7,14)(6,12,17,18)(8,13,16)(0,10,11)(1,3,5)$
round 4: $(5,8,17)(3,6,16)(7,10,15,18)(9,11,14)(1,2,4)(0,12,13)$
round 5: $(2,11,16)(4,10,17)(5,9,13,18)(3,7,12)(0,14,15)(1,6,8)$
round 6: $(3,13,14)(5,12,15)(4,8,11,18)(2,6,10)(1,7,9)(0,16,17)$

Our final example involves combining two SGA solutions for sets of v_1 and v_2 points respectively, each with block sizes k and $k+1$, to create solutions for $SGA(v, \{k, k+1\}, r)$ for $v = v_1 + v_2$ for suitable values of r. We only suggest using this approach for large values of v (see Section 5.4) but use a smaller example to demonstrate the approach.

Example 5. *We can construct a solution for* $SGA(38, \{3,4\}, 6)$ *by combining two copies of the solution for* $SGA(19, \{3,4\}, 6)$ *constructed in Example 4. Consider the sets* $V_1 = \{0, 1, 2, \ldots, 17, 18\}$ *and* $V_2 = \{0', 1', 2', \ldots, 18'\}$ *and combine the allocations on* $V_1 \cup V_2$ *as follows:*

round 1: $(0,2,3,18)(1,15,17)(5,6,11)(4,7,13)(8,12,14)(9,10,16)$
$(0',2',3',18')(1',15',17')(5',6',11')(4',7',13')(8',12',14')(9',10',16')$
round 2: $(1,4,16,18)(0,4,5)(3,8,10)(2,9,12)(6,13,15)(7,11,17)$
$(1',4',16',18')(0',4',5')(3',8',10')(2',9',12')(6',13',15')(7',11',17')$
round 3: $(4,9,15)(2,7,14)(6,12,17,18)(8,13,16)(0,10,11)(1,3,5)$
$(4',9',15')(2',7',14')(6',12',17',18')(8',13',16')(0',10',11')(1',3',5')$
round 4: $(5,8,17)(3,6,16)(7,10,15,18)(9,11,14)(1,2,4)(0,12,13)$
$(5',8',17')(3',6',16')(7',10',15',18')(9',11',14')(1',2',4')(0',12',13')$
round 5: $(2,11,16)(4,10,17)(5,9,13,18)(3,7,12)(0,14,15)(1,6,8)$
$(2',11',16')(4',10',17')(5',9',13',18')(3',7',12')(0',14',15')(1',6',8')$
round 6: $(3,13,14)(5,12,15)(4,8,11,18)(2,6,10)(1,7,9)(0,16,17)$
$(3',13',14')(5',12',15')(4',8',11',18')(2',6',10')(1',7',9')(0',16',17')$

A solution for the set $V = \{0, 1, 2, \ldots, 33, 34, 37\}$ *can then be obtained by renaming* $0'$ *as* 19, $1'$ *as* 20, $2'$ *as* 21 *and so on:*

round 1: $(0,2,3,18)(1,15,17)(5,6,11)(4,7,13)(8,12,14)(9,10,16)$
$(19,21,22,37)(20,34,36)(24,25,30)(23,26,32)(27,31,33)(28,29,35)$
round 2: $(1,4,16,18)(0,4,5)(3,8,10)(2,9,12)(6,13,15)(7,11,17)$
$(20,23,35,37)(19,23,24)(22,27,29)(21,28,31)(25,32,34)(26,30,36)$
round 3: $(4,9,15)(2,7,14)(6,12,17,18)(8,13,16)(0,10,11)(1,3,5)$
$(23,28,34)(21,26,33)(25,31,36,37)(27,32,35)(19,29,30)(20,22,24)$
round 4: $(5,8,17)(3,6,16)(7,10,15,18)(9,11,14)(1,2,4)(0,12,13)$
$(24,27,36)(22,25,35)(26,29,34,37)(28,30,33)(20,21,23)(19,31,32)$
round 5: $(2,11,16)(4,10,17)(5,9,13,18)(3,7,12)(0,14,15)(1,6,8)$
$(21,30,35)(23,29,36)(24,28,32,37)(22,26,31)(19,33,34)(20,25,27)$
round 6: $(3,13,14)(5,12,15)(4,8,11,18)(2,6,10)(1,7,9)(0,16,17)$
$(22,32,33)(24,31,34)(23,27,30,37)(21,25,29)(20,26,28)(19,35,36)$

In the above example, the two smaller sets had equal size—this need not be the case. If we were trying to find an allocation where v is odd, we would choose smaller sets of sizes $(v-1)/2$ and $(v+1)/2$ and use allocations for the smaller sets for which the blocks have similar sizes.

In the remainder of this section, we will examine $SGA(v,K,r)$ solutions, for which the following conditions hold:

1. $12 \leq v \leq 50$,
2. there are at most 2 block sizes,
3. block sizes differ by at most 1,
4. every round has the same distribution of block sizes, i.e., either all rounds have blocks of the same size (k) or they have m_1 blocks of size k and m_2 blocks of size $k+1$, where m_1 and m_2 are the same for all rounds
5. If $v \geq 20$ and there are two block sizes then all blocks have size between 4 and 6.
6. r is the largest for which a solution has been found.

The reason for our focus on this constrained set of solutions to $SGA(v,K,r)$ is that they correspond to good, balanced allocation schedules (see Definition 1).

We let K be the set of block sizes, where $K=\{k\}$ (a set containing a single value k) or $K=\{k,k+1\}$ (a set containing values k and $k+1$). If $K=\{k\}$ then, in order for there to be at least two rounds, we must have $v \geq k^2$. Similarly if $K=\{k,k+1\}$, then $v \geq k^2+k+1$.

In this paper, we present solutions to $SGA(v,K,r)$ for all $v \leq 50$, where K satisfies the conditions above, and where there are m_1 blocks of size k and m_2 blocks of size $k+1$. We will use the notation described in Table 1. Note that the best available solution for $SG(v,k)$ is a published solution for $SG(v,k,r)$, for which r is largest.

Table 1. Notation used in Tables.

Notation	Description
$KS(v,k)$	use a $KS(v,k)$
$KS(v,k)-p$	use a $KS(v,k)$ with p points removed from a single block in the final round
$KS(v,k)-p,B$	use a $KS(v,k)$ with p points removed from a single block in the final round and the final round of blocks removed
$NKTS(v)$	use a $NKTS(v)$
$RTD(k,n)$	use the blocks of an $RTD(k,n)$
$RTD(k,n)-p$	use the blocks of an $RTD(k,n)$ with p points removed from a single group
$SG(v,k)$	use the best available solution to the social golfer problem for v players in groups of size k
$SG(v,k)-p$	use the best available solution to the social golfer problem for v players in groups of size k, with p points removed, no pair of which appear in a block of the $SG(v,k)$
$SG(v,k)-p,B$	as above, but remove the points from final block and remove the final round of blocks
$D+p$	use an existing design D (a Kirkman system or resolvable transversal design, say), with p points added, each to a single block in each round, where these blocks don't intersect, for as many rounds as possible

All of our solutions are presented in Tables 2–4. For brevity, we denote sets $\{k\}$ and $\{k,k+1\}$ as k and $k,k+1$, respectively, and for any empty cell, refer to the values in the cell above. We show the theoretical maximum number of rounds (MAX), where MAX is determined from the results of Lemma 3. We indicate cases where Lemma 3 implies that $MAX < R(v,k,m_1,m_2)$ with an asterisk. This maximum is not necessarily achievable. In the last two columns, we indicate how a set of parallel rounds for these parameters can be obtained, using existing results and techniques similar to those used in Examples 3 and 4 and show the actual number of rounds achieved (r).

For any set of parameters v, k, m_1 and m_2, many solutions may exist—we only provide one solution, as that is all we need for our purposes. Listing all solutions and classifying them would be a different problem and out of the scope of this paper. For similar reasons, although r may be the best possible number of rounds in some instances, we do not claim this to be true in all cases, or attempt to prove it when it is. We simply include the largest solution that we have found so far.

Table 2. Solutions to $SGA(v,K,r)$ where $r \geq 3$, $12 \leq v \leq 19$, and $K = \{3\}, \{4\}$ or $\{3,4\}$.

v	K	$m1, m2$	MAX	Solution	r
12	3	4, 0	4*	$KS(16,4)-4,B$	4
13	3, 4	3, 1	5	$KS(16,4)-3,B$	4
14	3, 4	2, 2	5	$KS(16,4)-2,B$	4
15	3	5, 0	7	$KS(15,3)$	7
	3, 4	1, 3	5	$KS(16,4)-1$	5
16	4	4, 0	5	$KS(16,4)$	5
	3, 4	4, 1	6	$RTD(4,5)-4$	5
17	3, 4	3, 2	6	$RTD(4,5)-3$	5
18	3	6, 0	8	$NKTS(18,3)$	8
	3, 4	2, 3	6	$RTD(4,5)-2$	5
19	3, 4	1, 4	6	$RTD(4,5)-1$	5
		5, 1	8	$NKTS(18)+1$	6

Table 3. Solutions to $SGA(v, K, r)$ where $r \geq 3$, $20 \leq v \leq 40$, and $K = \{3\}, \{4\}, \{5\}, \{4,5\}$ or $\{5,6\}$.

v	K	$m1, m2$	MAX	Solution	r
20	4	5, 0	5*	$RTD(4,5)$	5
21	3	7, 0	10	$KS(21,3)$	10
	4, 5	4, 1	6	$KS(25,5)-4,B$	5
22	4, 5	3, 2	6	$KS(25,5)-3,B$	5
23	4, 5	2, 3	6	$KS(25,5)-2,B$	5
24	3	8, 0	11	$NKTS(24,3)$	11
	4	6, 0	7	$KS(24,4)$	7
	4, 5	1, 4	6	$KS(25,5)-1$	6
25	5	5, 0	6	$KS(25,5)$	6
	4, 5	5, 1	7	$SG(30,5)-5,B$	5
26	4, 5	4, 2	7	$SG(30,5)-4,B$	5
27	3	9, 0	13	$KS(27,3)$	13
	4, 5	3, 3	7	$SG(30,5)-3,B$	5
28	4	7, 0	9	$KS(28,4)$	9
	4, 5	2, 4	7	$SG(30,5)-2$	6
29	4, 5	1, 5	7	$SG(30,5)-1$	6
		6, 1	8	$RTD(5,7)-6$	7
30	3	10, 0	14	$NKTS(30,3)$	14
	5	6, 0	7	$SG(30,5)$	6
	4, 5	5, 2	8	$RTD(5,7)-5$	7
31	4, 5	4, 3	8	$RTD(5,7)-4$	7
	5, 6	5, 1	7	$SG(36,6)-5$	3
32	4	8, 0	10	$SG(32,4)$	10
	4, 5	3, 4	8	$RTD(5,7)-3$	7
	5, 6	4, 2	7	$SG(36,6)-4$	3
33	3	11, 0	16	$KS(33,3)$	16
	4, 5	2, 5	8	$RTD(5,7)-2$	7
		7, 1	10	$RTD(5,8)-7$	8
	5, 6	3, 3	7	$SG(36,6)-3$	3
34	4, 5	1, 6	8	$RTD(5,7)-1$	7
		6, 2	10	$RTD(5,8)-6$	8
	5, 6	2, 4	7	$SG(36,6)-2$	3
35	5	7, 0	8	$RTD(5,7)$	7
	4, 5	5, 3	9	$RTD(5,8)-5$	8
	5, 6	1, 5	7	$SG(36,6)-1$	3
36	3	12, 0	17	$NKTS(36,3)$	17
	4	9, 0	11	$SG(36,4)$	8
	6	6, 0	3*	$SG(36,6)$	3
	4, 5	4, 4	9	$RTD(5,8)-4$	8
	5, 6	6, 1	8	$RTD(6,7)-6$	7
37	4, 5	3, 5	9	$RTD(5,8)-3$	8
		8, 1	11	$RTD(5,9)-8$	9
	5, 6	5, 2	8	$RTD(6,7)-5$	7
38	4, 5	2, 6	9	$RTD(5,8)-2$	8
		7, 2	11	$RTD(5,9)-7$	9
	5, 6	4, 3	8	$RTD(6,7)-4$	7
39	3	13, 0	19	$KS(39,3)$	19
	4, 5	1, 7	9	$RTD(5,8)-1$	8
		6, 3	11	$RTD(5,9)-6$	9
	5, 6	3, 4	8	$RTD(6,7)-3$	7
40	4	10, 0	13	$KS(40,4)$	13
	5	8, 0	9	$RTD(5,8)$	8
	4, 5	5, 4	11	$RTD(5,9)-5$	9
	5, 6	2, 5	8	$RTD(6,7)-2$	7

Table 4. Solutions to $SGA(v, K, r)$ where $r \geq 3$, $41 \leq v \leq 50$ and $K = \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{4,5\}$ or $\{5,6\}$.

v	K	$m1, m2$	MAX	Solution	r
41	4,5	4,5	11	$RTD(5,9)-4$	9
		9,1	12	$KS(40,4)+1$	9
	5,6	1,6	8	$RTD(6,7)-1$	7
		7,1	9	$RTD(6,8)-7$	8
42	3	14,0	20	$NKTS(42,3)$	20
	6	7,0	8	$RTD(6,7)$	7
	4,5	3,6	11	$RTD(5,9)-3$	9
		8,2	12	$KS(40,4)+2$	6
	5,6	6,2	9	$RTD(6,8)-6$	8
43	4,5	2,7	11	$RTD(5,9)-2$	9
		7,3	12	$KS(40,4)+3$	5
	5,6	5,3	9	$RTD(6,8)-5$	8
44	4	11,0	14	$RTD(5,11)-11$	11
	4,5	1,8	11	$RTD(5,9)-1$	9
		6,4	12	$SG(50,5)-6$	7
	5,6	4,4	9	$RTD(6,8)-4$	8
45	3	15,0	22	$KS(45,3)$	22
	5	9,0	11	$RTD(5,9)$	9
	4,5	5,5	12	$SG(50,5)-5$	7
		10,1	14	$RTD(5,11)-10$	11
	5,6	3,5	9	$RTD(6,8)-3$	8
46	4,5	4,6	12	$SG(50,5)-4$	7
		9,2	13	$RTD(5,11)-9$	11
	5,6	2,6	9	$RTD(6,8)-2$	8
		8,1	10	$RTD(6,9)-8$	9
47	4,5	3,7	12	$SG(50,5)-3$	7
		8,3	13	$RTD(5,11)-8$	11
	5,6	1,7	9	$RTD(6,8)-1$	8
		7,2	10	$RTD(6,9)-7$	9
48	3	16,0	23	$NKTS(48,3)$	23
	4	12,0	15	$RTD(4,12)$	12
	6	8,0	9	$RTD(6,8)$	8
	4,5	2,8	12	$SG(50,5)-2$	7
		7,4	13	$RTD(5,11)-7$	11
	5,6	6,3	10	$RTD(6,9)-6$	9
49	7	7,0	8	$KS(49,7)$	8
	4,5	1,9	12	$SG(50,5)-1$	7
		6,5	13	$RTD(5,11)-6$	11
		11,1	15	$RTD(5,12)-11$	12
	5,6	5,4	10	$RTD(6,9)-5$	9
50	5	10,0	12	$SG(50,5)$	7
	4,5	5,6	13	$RTD(5,11)-5$	11
		10,2	15	$RTD(5,12)-10$	12
	5,6	4,5	10	$RTD(6,9)-4$	9

5. Good, Balanced Allocation Schedules

In this section, we return to our original question: how to create good, balanced allocation schedules. Specifically, we want to create allocation schedules that satisfy the conditions of Definition 1. All of the allocation schedules can be obtained from our website (see Section 6) for as many rounds as required, up to the maximum number considered.

5.1. At Most Eleven Participants

When the number of participants v is smaller than 12, then, unless $v = 9$, we cannot create allocations without allowing groups of size 2. If v is 3 or 5, there is no solution unless some groups have size 1, which contradicts the point of using groups, so we ignore those cases. The case $v = 4$, with 2 groups of size 2 for 3 rounds, is trivial. A solution for each of the cases $6 \leq v \leq 12$ is obtained by considering $SGA(v, K, r)$ solutions, using the techniques described in Section 4.1 (but in this case, relaxing the constraints on group (block) sizes). For each value of v, we consult Table 5 below. For example, for $v = 6$, we can only have 3 groups of size 2 in each round of our allocation schedule, because any allocation schedule must match up to a design listed in the table. From the table we see that such an allocation is available, how it is constructed (using a solution available for $SG(6, 2)$) and that the solution provides an allocation schedule consisting of 5 rounds.

Allocation schedules for other values of $v \leq 11$ are obtained in a similar way.

Table 5. Solutions to $SGA(v, K, r)$ where $r \geq 3$, $6 \leq v < 12$ and $K = \{2\}, \{3\}$ or $\{2, 3\}$.

v	K	$m1, m2$	MAX	Solution	r
6	2	3, 0	5	$SG(6, 2)$	5
7	2, 3	2, 1	4	$KS(9, 3) - 2, B$	3
8	2	4, 0	7	$SG(8, 2)$	7
	2, 3	1, 2	4	$KS(9, 3) - 1$	4
9	3	3, 0	4	$KS(9, 3)$	4
	2, 3	3, 1	6	$SG(8, 2) + 1$	4
10	2	5	9	$SG(10, 2)$	9
	2, 3	2, 2	5	$SG(8, 2) + 2$	4
11	2, 3	1, 3	5	$KS(16, 4) - 5, B$	4
		4, 1	7	$SG(10, 2) + 1$	5

5.2. Between Twelve and Nineteen Participants

Now, we can insist that groups have a size of at least 3. In all cases, groups must have size 3 or 4 and we can obtain solutions up to r rounds by considering the solutions to $SGA(v, K, r)$ indicated in Table 2.

5.3. Between Twenty and Fifty Participants

For all values of $v \geq 20$, there is at least one allocation schedule for which all groups have a size of at least 4. For this reason, we no longer consider groups of size 3 unless 3 divides the number of participants, and there is an allocation schedule for which all blocks have size 3. Good, balanced allocation schedules can be obtained from the $SGA(v, K, r)$ solutions listed in Tables 3 and 4.

5.4. More than Fifty Participants

It would be tempting to continue constructing tables of solutions for larger and larger numbers of participants. Indeed, we use solutions constructed in this way for allocation schedules provided by our web-based tool (see Section 6). These currently include all good balanced allocations for up to 70 participants and some for larger numbers of participants. However, generating solutions in this way for large numbers of participants is not necessary. We do not, after all, require a best possible solution (in terms of number of rounds), rather, we need a single solution for which a reasonable number of rounds is possible. A simpler solution is to divide the participants into two roughly equal sized cohorts. We can then glue together allocations for the two smaller numbers of participants to achieve an allocation for all participants. We cannot guarantee that there will be only two block sizes in this case, but we can try to minimise the number of different block sizes.

Example 5 in Section 4.1 illustrates this approach.

6. Obtaining Solutions from Our Website

Over 270 balanced allocations for between 6 and 85 participants are currently indexed on our website http://www.dcs.gla.ac.uk/~alice/. This includes all good, balanced allocation schedules for up to 70 participants and some additional schedules for larger numbers of participants. We will continue to add solutions as they are generated. Visitors can specify the number of participants and the number of rounds and all corresponding allocation schedules are shown. The allocations will have varying group sizes. They are displayed as comma-separated lists which can be exported as plain-text. Alternatively, users can provide a list of email addresses to translate the allocations for their participants. In Zoom, when setting the breakout room assignments, hosts can import a list of email addresses to automatically assign participants. When a list of usernames is imported to our website, a file suitably formatted for assignment on Zoom is provided.

7. Discussion

We have identified a topical problem of allocating participants to breakout rooms when using online meeting software solutions such as Zoom and Microsoft Teams. Specifically, we have investigated how to schedule sequences of rounds in a balanced way, whereby no two participants appear in a group together more than once. By additionally imposing the constraints that group sizes should differ by at most one and each round should have the same distribution of group sizes, we have defined a new combinatorial structure, namely a variant of the social golfer problem (SGP), which we have called the social golfer problem with adjacent group sizes (SGA).

In the context of breakout rooms, groups of sizes 4–6 are preferable, so we have imposed additional constraints in this instance, and generated solutions for as many rounds as we can, using either available solutions for SGP, known constructions for Kirkman systems, nearly Kirkman systems and resolvable transversal designs, or by adding or removing points from such structures. We have provided an online resource so that our solutions can be available to all.

There is plenty of scope for additional contribution to the investigation of SGA. Outside of the particular application of breakout rooms, there is no restriction on the particular sizes of the groups (as long as the other properties are satisfied). For example, for $v = 43$, there are SGA solutions for group sizes 6 and 7 (where each round has 6 groups of size 6 and one of size 7), which we do not consider here. In addition, our methods can be extended for increasing numbers of participants, and for any SGA instance, the upper bound on the number of rounds possible could be refined, often by using simple counting methods. Solutions with a number of rounds closer to the upper bound could be found using more specific constructions or instances from combinatorial design theory (e.g., constructions for class-uniformly resolvable pairwise balanced designs with two block sizes, and pairwise balanced designs [52,53]). Alternatively, constraint programming or SAT solving techniques, such as those used to find maximal solutions to SGP could be used to find solutions with more rounds.

The SGP has many practical applications, such as encoding, encryption and covering problems, as well as a wide range of scheduling problems. Our novel concept of SGA will allow these applications to be extended to situations in which unequal group sizes are required.

Our allocation schedules treat all participants equally—i.e., we do not consider ability or roles within a classroom (we cannot insist, for example, that an A-grade student is present in every group, or that no two struggling students appear in a group together). The composition of groups is important [54] and there is evidence to suggest that mixed ability groups are associated with positive educational and developmental outcomes [55,56], but we acknowledge that the opposite may also be the case [57], and that it may be advantageous to assign a student with particular learning needs to a particular group. It would be possible to modify our approach to accommodate these additional constraints, but this is left as future work.

Author Contributions: Writing—original draft preparation, A.M., M.B., W.K.; writing—review and editing, A.M., M.B., W.K., I.V. and H.C.P.; Methodology, A.M., M.B.; software, A.M., W.K., I.V.; Data curation, I.V., Investigation, A.M., M.B., W.K., I.V., H.C.P. All authors have read and agreed to the published version of the manuscript.

Funding: William Kavanagh and Ivaylo Valkov were supported by the EPSRC Doctoral Training Partnership award EP/M508056/1 and EPSRC grant EP/N007565/1 respectively.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chase, J.D.; Okie, E.G. Combining cooperative learning and peer instruction in introductory Computer Science. In Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education (SIGCSE '00), Austin, TX, USA, 8–12 March 2000; ACM: New York, NY, USA, 2000; pp. 372–376.
2. Laal, M.; Ghodsi, S.M. Benefits of collaborative learning. *Procedia Soc. Behav. Sci.* **2012**, *31*, 486–490. [CrossRef]
3. Gokhale, A.A. Collaborative learning enhances critical thinking. *J. Technol. Educ.* **1995**. . [CrossRef]
4. Kreijns, K.; Kirschner, P.A.; Jochems, W. Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: A review of the research. *Comput. Hum. Behav.* **2003**, *19*, 335–353. [CrossRef]
5. Mendo-Lázaro, S.; León-del Barco, B.; Felipe-Castaño, E.; Polo-del Río, M.; Iglesias-Gallego, D. Cooperative Team Learning and the Development of Social Skills in Higher Education: The Variables Involved. *Front. Psychol.* **2018**, *9*, 1536. [CrossRef] [PubMed]
6. Akçayir, G.; Akçayir, M. The flipped classroom: A review of its advantages and challenges. *Comput. Educ.* **2018**, *126*, 334–345. [CrossRef]
7. Bishop, J.L.; Verleger, M.A. The flipped classroom: A survey of the research. In Proceedings of the ASEE National Conference Proceedings, Atlanta, GA, USA, 23–26 June 2013; Volume 30, pp. 1–18.
8. Bhagat, K.K.; Chang, C.N.; Chang, C.Y. The impact of the flipped classroom on mathematics concept learning in high school. *J. Educ. Technol. Soc.* **2016**, *19*, 134–142.
9. Wanner, T.; Palmer, E. Personalising learning: Exploring student and teacher perceptions about flexible learning and assessment in a flipped university course. *Comput. Educ.* **2015**, *88*, 354–369. [CrossRef]
10. Eichler, J.F.; Peeples, J. Flipped classroom modules for large enrollment general chemistry courses: A low barrier approach to increase active learning and improve student grades. *Chem. Educ. Res. Pract.* **2016**, *17*, 197–208. [CrossRef]
11. Ben-Ari, M. Constructivism in Computer Science Education. *J. Comput. Math. Sci. Teach.* **2001**, *20*, 45–73.
12. Porter, L.; Bailey Lee, C.; Simon, B.; Zingaro, D. Peer instruction: Do students really learn from peer discussion in computing? In Proceedings of the Seventh International Workshop on Computing Education Research (ICER '11), Providence, RI, USA, 8–9 August 2011; ACM: New York, NY, USA, 2011; pp. 45–52.
13. Porter, L.; Bailey, L.; Simon, B. Halving fail rates using peer instruction: A study of four Computer Science courses. In Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE'13), Denver, CO, USA, 6–9 March 2013; ACM: New York, NY, USA, 2013; pp. 177–182.
14. Simon, B.; Kohanfars, M.; Lee, J.; Tamayo, K.; Cutts, Q. Experience report: Peer instruction in introductory computing. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10), Milwaukee, WI, USA, 10–13 March 2010; ACM: New York, NY, USA, 2010; pp. 341–345.
15. Simon, B.; Parris, J.; Spacco, J. How we teach impacts student learning: Peer instruction vs. lecture in CS0. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE'13), Denver, CO, USA, 6–9 March 2013; ACM: New York, NY, USA, 2013; pp. 41–46.
16. Porter, L.; Bouvier, D.; Cutts, Q.; Grissom, S.; Lee, C.; McCartney, R.; Zingaro, D.; Simon, B. A Multi-institutional Study of Peer Instruction in Introductory Computing. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE'16), Memphis, TN, USA, 2–5 March 2016; ACM: New York, NY, USA, 2016; pp. 358–363.
17. Lee, C.B.; Garcia, S.; Porter, L. Can peer instruction be effective in upper-division Computer Science courses? *ACM Trans. Comput. Educ.* **2013**, *13*, 12:1–12:22. [CrossRef]
18. Barr, M.; Nabi, S.W.; Somerville, D. Online Delivery of Intensive Software Engineering Education During the COVID-19 Pandemic. In Proceedings of the 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T), Munich, Germany, 9–12 November 2020.
19. Keller, R.T. Predictors of the Performance of Project Groups in R & D Organizations. *Acad. Manag. J.* **1986**, *29*, 715–726.
20. Gibbs, G. *The Assessment of Group Work: Lessons from the Literature*; *Assessment Standards Knowledge Exchange*; Oxford Brookes University: Oxford, UK, 2009; pp. 1–17.
21. Kooloos, J.; Klaassen, T.; Vereijken, M.; Van Kuppeveld, S.; Bolhuis, S.; Vorstenbosch, M. Collaborative group work: Effects of group size and assignment structure on learning gain, student satisfaction and perceived participation. *Med Teach.* **2011**, *33*, 983–988. [CrossRef] [PubMed]
22. Colbourn, C.; Dinitz, J. (Eds.) *Handbook of Combinatorial Designs*, 2nd ed.; CRC Press: New York, NY, USA, 2007.

23. Smith, B. Reducing Symmetry in a Combinatorial Design problem. In Proceedings of the Third International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'01), Kent, UK, 8–10 April 2001; pp. 351–359.
24. Focacci, F.; Milano, M. Global Cut Frame-work for Removing Symmetries. In Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP 2001), Paphos, Cyprus, 26 November–1 December 2001; pp. 75–92.
25. Petrie, K.; Smith, B.; Yorke-Smith, N. Dynamic symmetry breaking in constraint programming and linear programming hybrids. In Proceedings of the European Starting AI Researcher Symposium, Valencia, Spain, 23–24 August 2004.
26. Donaldson, A.F.; Miller, A.; Calder, M. Comparing the use of symmetry in constraint processing and model checking. In Proceedings of the 4th International Workshop on Symmetry and Constraint Satisfaction Problems, Toronto, ON, Canada, 27 September 2004; pp. 18–25.
27. Gent, I.; Kelsey, T.; Linton, S.; McDonald, I.; Miguel, I.; Smith, B. Conditional Symmetry Breaking. In Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP 2005), Sitges, Spain, 1–5 October 2005; pp. 256–270.
28. Gent, I.P.; Petrie, K.; Puget, J.F. *Handbook of Constraint Programming, Chapter Symmetry in Constraint Programming*; Elsevier: Oxford, UK, 2006.
29. Triska, M. Solution Methods for the Social Golfer Problem. Master's Thesis, Technische Universität Vienna, Vienna, Austria, 2008.
30. Harvey, W. CSPLib Problem 010: Social Golfers Problem. Available online: http://www.csplib.org/Problems/prob010 (accessed on 12 December 2020).
31. Barnier, N.; Brisset, P. Solving the Kirkman's Schoolgirl Problem in a Few Seconds. In Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP 2002), Ithaca, NY, USA, 9–13 September 2002; pp. 477–491.
32. Dotú, I.; Van Hentenryck, P. Scheduling social golfers locally. In *Lecture Notes in Computing Science, Proceedings of the 2nd International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Prague, Czech Republic, 31 May–1 June 2005*; Springer Berlin, Germany, 2005; Volume 3524, pp. 155–167.
33. Gent, I.; Lynce, I. A SAT encoding for the social golfer problem. In Proceedings of the IJCAI'05 Workshop on Modelling and Solving Problems with Constraints, Edinburgh, Scotland, 30 July–5 August 2005.
34. Triska, M.; Musliu, N. An improved SAT formulation for the social golfer problem. *Ann. Oper. Res.* **2010**, *194*, 427–438. [CrossRef]
35. Law, Y.C.; Lee, J.H. Global Constraints for Integer and Set Value Precedence. In Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP 2004), Toronto, ON, Canada, 27 September–1 October 2004; pp. 362–376.
36. Liu, K.; Löffler.; Hofstedt, P. Solving the Social Golfers Problems by Constraint Programming in Sequential and Parallel. In Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019), Prague, Czech Republic, 19–21 February 2019; Volume 2, pp. 29–39.
37. Triska, M.; Musliu, N. An effective greedy heuristic for the Social Golfer Problem. *Ann. Oper. Res.* **2012**, *194*, 413–425. [CrossRef]
38. Higazy, M.; El-Mesady, A.; Mohammed, M.S. On Graph-Orthogonal Arrays by Mutually Orthogonal Graph Squares. *Symmetry* **2020**, *12*, 1895. [CrossRef]
39. Kirkman, T. On a problem in combinatorics. *Camb. Dublin Math. J.* **1847**, *2*, 191–204.
40. Ray-Chaudhuri, D.; Wilson, R. Solution of Kirkman's school girl problem. *Proc. Symp. Pure Math.* **1971**, *19*, 187–203.
41. Harvey, W.; Winterer, T. Solving the MOLR and Social Golfers Problems. In *Lecture Notes in Computing Science, Proceedings of Constraints Programming (CP 2005), Sitges, Spain, 1–5 October 2005*; Springer: Berlin, Germany, 2005; Volume 3709, pp. 286–300.
42. Pegg, E. Math Games: Social Golfer Problem. Available online: http://www.mathpuzzle.com/MAA/54-Golf%20Tournaments/mathgames_08_14_07.html (accessed on 14 December 2020).
43. Harvey, W. CSPLib Problem 010: Social Golfers Problem, Results. Available online: http://www.csplib.org/Problems/prob010/results/ (accessed on 14 December 2020).
44. Zhu, L. Some recent developments on BIBDs and related designs. *Discret. Math.* **1993**, *123*, 189–214. [CrossRef]
45. Furino, S.; Miao, Y.; Yin, J. *Frames and Resolvable Designs*; CRC Press: Boca Raton, FL, USA, 1996.
46. Kageyama, S. A Survey of Reolvable Solutions of Balanced Incomplete Block Designs. *Int. Stat. Rev.* **1972**, *40*, 269–273. [CrossRef]
47. Colbourn, C.; Kaskib, P.; Östergård.; Pike, D.; Pottonen, O. Nearly Kirkman triple systems of order 18 and Hanani triple systems of order 19. *Discret. Math.* **2011**, *311*, 827–834. [CrossRef]
48. Abel, R.; Chan, N.; Colbourn, C.; Lamken, E.; Wang, C.; Wang, J. Doubly Resolvable Nearly Kirkman Triple Systems. *J. Comb. Des.* **2013**, *21*, 342–358. [CrossRef]
49. Baker, R.; Wilson, R. Nearly Kirkman Triple Systems. *Util. Math.* **1977**, *11*, 289–296.
50. Kreher, D.; Ling, A.; Rees, R.; Lam, C. A note on $\{4\}$-GDDs of type 2^{10}. *Discret. Math.* **2003**, *261*, 373–376. [CrossRef]
51. Bose, R.; Shrikhande, S.S.; Parker, E. Further Results on the Construction of Mutually Orthogonal Latin Squares and the Falsity of Euler's Conjecture. *Can. J. Math.* **1960**, *12*, 189–203. [CrossRef]
52. Lamken, E.; Rees, R.; Vanstone, S. Class-uniformly resolvable pairwise balanced designs with block sizes two and three. *Discret. Math.* **1991**, *92*, 197–209. [CrossRef]
53. Dukes, P.; Lamken, E. Constructions and uses of incomplete pairwise balanced designs. *Des. Codes Cryptogr.* **2019**, *87*, 2729–2751. [CrossRef]
54. Burini, D.; De Lillo, S. On the Complex Interaction between Collective Learning and Social Dynamics. *Symmetry* **2019**, *11*, 967. [CrossRef]

55. Burris, C.C.; Wiley, E.; Welner, K.; Murphy, J. Accountability, rigor, and detracking: Achievement effects of embracing a challenging curriculum as a universal good for all students. *Teach. Coll. Rec.* **2008**, *110*, 571–607.
56. Ireson, J.; Hallam, S. *Ability Grouping in Education*; Sage: London, UK, 2001
57. Steenbergen-Hu, S.; Makel, M.C.; Olszewski-Kubilius, P. What one hundred years of research says about the effects of ability grouping and acceleration on K–12 students' academic achievement: Findings of two second-order meta-analyses. *Rev. Educ. Res.* **2016**, *86*, 849–899. [CrossRef]