Parallel-Class Splitting and Constructive Lower Bounds for the Social Golfer Problem

Guido Witt-Dörring

August 2026

Abstract

We establish constructive lower bounds for the Social Golfer Problem using resolvable designs and explicit schedules. We show that, for $p \geq 2$, any index-one RTD$(p, 2p-1)$ can have one parallel class replaced by two compatible classes, yielding $W(2p-1, p) \geq 2p$. In particular, this holds whenever $2p-1$ is a prime power. Given an RTD$(p, 2p)$, the same replacement works when the number of groups is $g = 2p$, although the resulting lower bound already follows from standard group filling. We also derive a ten-round schedule for groups of five on 50 players by completing a partial class of Brouwer's ITD$(10, 2; 6)$ with two hole transversals. Finally, cyclic difference arrays or complete schedules establish $W(20, 7) \geq 15$, $W(21, 7) \geq 16$, $W(28, 7) \geq 21$, and $W(14, 13) \geq 6$. No exactness or optimality claim is made.

1 Introduction

The Social Golfer Problem asks how long a fixed population can be repartitioned into equal groups without repeating a pair of group-mates. Its schedules are resolvable packings. Latin-square and transversal-design constructions give families of schedules, and finite cases serve as benchmarks for constraint programming and heuristic search [8, 11].

The main contributions are as follows.

1. We give a parallel-class splitting construction for an index-one RTD$(p, 2p-1)$, replacing one class by two compatible classes and proving $W(2p-1, p) \geq 2p$.

2. We complete a partial class of Brouwer's ITD$(10, 2; 6)$ with two transversals of its hole, obtaining $W(10, 5) \geq 10$.

3. We give cyclic difference arrays or complete schedules for four further lower bounds:

$$W(20, 7) \geq 15, \quad W(21, 7) \geq 16, \quad W(28, 7) \geq 21, \quad W(14, 13) \geq 6.$$

In a targeted search completed September 7, 2026, we did not locate an equivalent odd-parameter RTD splitting statement or an earlier publication of the five displayed finite bounds. The search covered the exact parameters and RTD/TD, MOLS/MOLR, resolvable-packing, difference-structure, trade, switching, and generalized balanced tournament-design terminology. The search was not exhaustive and does not establish publication priority. The results are constructive lower bounds; no exactness or optimality claim is made.

Section 2 fixes notation. Section 3 describes the relevant existing literature. Section 4 gives the constructions and proofs, section 5 states the finite bounds, and section 6 gives the finite verification procedure.

2 Definitions

Let g be the number of groups in each round, let p be the group size, and let V be a player set of size gp. A schedule for $g{-}p{-}w$ is a sequence $\mathcal{P}_1, \ldots, \mathcal{P}_w$ in which each $\mathcal{P}_t$ partitions V into g blocks of size p. The schedule is *zero-repeat* if every pair of players occurs together in at most one block. We write

$$W(g,p) = \max\{w : \text{a zero-repeat schedule for } g{-}p{-}w \text{ exists}\}.$$

The elementary partner-counting bound is

$$W(g,p) \le \left\lfloor \frac{gp-1}{p-1} \right\rfloor. \tag{1}$$

We use $W(g,p) \ge w$ only for a constructive lower bound; none of the finite results below is asserted to meet equation (1).

A transversal design $\mathrm{TD}(k,n)$ has k *design groups* of size n and blocks of size k, each block meeting each design group once, such that every pair from different design groups lies in exactly one block. An index-one resolvable transversal design $\mathrm{RTD}(k,n)$ has its blocks partitioned into n parallel classes. Equivalently, an $\mathrm{RTD}(k,n)$ can be obtained by deleting one design group from a $\mathrm{TD}(k+1,n)$; this is also equivalent to $k-1$ mutually orthogonal Latin squares of order n.

For $1 \le h < n$, an incomplete transversal design $\mathrm{ITD}(n,h;k)$ has k design groups of size n and a hole meeting each design group in h points. Pairs from different design groups occur once unless both points are in the hole, in which case they do not occur. This parameter order follows Miller et al. [8]; it is stated explicitly because conventions vary.

For a zero-repeat partial schedule, the *leave graph* has vertex set V and an edge for every pair not yet used. A K_p-factor of the leave is a partition of V into p-cliques and therefore represents one additional round.

3 Related work

SAT and heuristic formulations of the Social Golfer Problem address symmetry and the difficulty of completing highly constrained schedules [11]. Design-theoretic constructions obtain parallel classes from Latin squares, transversal designs, resolvable group-divisible designs, and related structures.

Harvey and Winterer [4] relate mutually orthogonal Latin rectangles (MOLR) to Social Golfer schedules and report computational lower bounds. Their construction uses the translation to resolvable incomplete block designs given by Sharma and Das [10]. Miller et al. [8] develop RTD, incomplete-RTD, difference-matrix, and group-filling constructions and tabulate schedules for up to 150 players. These works provide the principal published comparisons for the finite parameters in section 5.

The underlying objects in two of our constructions are known. Brouwer's four MOLS of order 10 with a common hole of order 2 give the $\mathrm{ITD}(10,2;6)$ used in proposition 5 [2]. Difference-matrix development into resolvable transversal designs is classical [5]. Likewise, filling latent groups and recognizing another round as a clique factor of the leave are standard operations.

Whenever an $\mathrm{RTD}(p,2p)$ exists, the even-parameter bound $W(2p,p) \ge 2p+1$ follows from the general group-filling construction of Miller et al. [8, Lemma 16 and Sec. 6.4]. Our parallel-class replacement gives another route to the same value, so we treat the even case as known and focus on the odd relation $g = 2p-1$.

Related column-partition constructions occur in Lamken's generalized balanced tournament designs [6]. A $\mathrm{GBTD}(n,k)$ is based on a design with pair multiplicity $k-1$, and its "partitioned"

condition partitions columns into prescribed sets. This differs from deleting one class of an index-one RTD packing and replacing it by two compatible classes.

4 Constructions

4.1 Splitting one parallel class of an RTD

Theorem 1 (Parallel-class splitting)**.** *Let $p \geq 2$, let $g = 2p - 1$, and suppose that an index-one* RTD(p, g) *exists. Then*

$$W(g, p) \geq g + 1.$$

Proof. Let the design groups be $H_1, \ldots, H_p$, and select one parallel class

$$\mathcal{C} = \{C_j : j \in J\}, \qquad |J| = g.$$

Because $\mathcal{C}$ partitions the points and every block meets every design group once, label

$$C_j = \{x_{1j}, \ldots, x_{pj}\}, \qquad x_{ij} \in H_i.$$

For a p-subset $S \subseteq J$, put

$$R_i(S) = \{x_{ij} : j \in S\}, \qquad \mathcal{P}(S) = \{R_i(S) : 1 \leq i \leq p\} \cup \{C_j : j \notin S\}.$$

The row blocks cover precisely the points in columns S, while the retained column blocks cover the complementary columns. Thus $\mathcal{P}(S)$ is a parallel class containing $p + (g - p) = g$ blocks of size p.

Choose p-subsets S, T so that $S \cup T = J$ and $|S \cap T| = 1$. Delete $\mathcal{C}$ and append $\mathcal{P}(S)$ and $\mathcal{P}(T)$.

A pair in a row block lies within one design group and therefore occurs in no transversal block. Every cross-design-group pair in a retained C_j occurred only in the deleted class. Hence neither replacement repeats a pair from an undeleted RTD class. A within-design-group pair could occur in both replacements only if two column indices belonged to $S \cap T$, which has size at most one. A cross-design-group pair could occur in both only in a column outside $S \cup T$, which does not exist. The two replacements are therefore compatible. Replacing one of the original g classes by two gives $g + 1$ rounds. □

The case $p = 1$ is excluded because singleton-block schedules are degenerate. The hypothesis can equivalently be stated as the existence of a TD$(p + 1, g)$ or of $p - 1$ mutually orthogonal Latin squares of order g. A plain TD(p, g) is not sufficient.

Corollary 2 (Prime-power instances)**.** *If $p \geq 2$ and $q = 2p - 1$ is a prime power, then*

$$W(q, p) \geq 2p.$$

Proof. The finite-field construction gives $q - 1$ mutually orthogonal Latin squares of order q. Select $p-1$ of them to obtain a TD$(p+1, q)$, equivalently an index-one RTD(p, q), and apply theorem 1. □

For this relation the elementary partner-count bound is $E(q, p) = 2p + 1$, so the corollary leaves at most one round unresolved. Table 1 distinguishes the earlier published comparisons in the finite table of Miller et al. [8] from the endpoint supplied by the transformation. In these three instances the split adds one round to that comparison; the remaining gap to the elementary ceiling is also one.

Remark 3 (Even parameter)**.** *Suppose that an* RTD$(p, 2p)$ *exists. The same replacement proof works with disjoint p-sets S, T whose union is J. Under this hypothesis, $W(2p, p) \geq 2p + 1$ already follows by retaining all RTD classes and adding one round obtained by partitioning the design groups, an instance of the group-filling construction in Miller et al. [8, Lemma 16 and Sec. 6.4].*

Table 1: Prime-power instances and selected published lower bounds.

p	q	Published lower	Split lower	E	Gap
5	9	9	10	11	1
6	11	11	12	13	1
7	13	13	14	15	1

4.2 Cyclic difference arrays

Let $D = (d_{i,t})$ be a $k \times n$ array over $\mathbb{Z}_m$. Use the player set

$$V = \{(i, x) : 0 \le i < k,\ x \in \mathbb{Z}_m\},$$

and, for column t and translation $a \in \mathbb{Z}_m$, define

$$B_{t,a} = \{(i, a + d_{i,t}) : 0 \le i < k\}. \tag{2}$$

Proposition 4 (Cyclic development)**.** *If, for every $i < j$, the n differences*

$$d_{i,t} - d_{j,t} \pmod m, \qquad 0 \le t < n,$$

are distinct, then equation (2) *gives $W(m, k) \ge n$.*

Proof. For fixed t, translation by $d_{i,t}$ is a bijection on every latent class $\{i\} \times \mathbb{Z}_m$, so the m blocks partition all km players. Players (i, x) and (j, y) from different latent classes meet in column t exactly when

$$x - y = d_{i,t} - d_{j,t} \pmod m.$$

Difference uniqueness allows at most one such column. Players in the same latent class do not meet in a developed block. □

Difference matrices and their connection with resolvable transversal designs are classical [5]. The at-most-once condition above is naturally viewed as a cyclic difference-packing condition; related terminology is developed by Yin [12] and Bailey and Burgess [1].

A further round exists exactly when the leave contains a K_k-factor. We use one simple special case. Developed rounds contain no within-latent-class pair. Therefore, whenever $k \mid m$, partitioning each of the k latent classes of size m into m/k blocks of size k supplies an additional round. This is an instance of standard resolvable group filling [3, 9].

4.3 Completing a holey transversal design

Proposition 5 (Hole completion)**.** *Let $1 \le h < n$ and $k \ge 3$. If an* ITD$(n, h; k)$ *exists and $k - 1$ divides n, then*

$$W(n, k - 1) \ge n - h + 2.$$

Proof. Delete one design group. The blocks through each of its $n - h$ non-hole points project to a complete parallel class on the surviving points. The blocks through a deleted hole point project to a partial class of $n - h$ blocks covering every surviving non-hole point.

The surviving hole consists of h points in each of the $k - 1$ remaining design groups. Arrange these points into h transversal blocks, each containing one point from every surviving design group. Pairs of hole points from different design groups occur in no ITD block, so these transversals

complete the partial class without a repeated pair. Finally, because $k-1 \mid n$, split every surviving design group into blocks of size $k-1$ and align all such blocks as one more parallel class. These within-design-group pairs were unused. We obtain $(n-h)+1+1$ rounds. □

For the finite application, Brouwer constructed a $T[6;10]-T[6;2]$, equivalently an ITD$(10,2;6)$ [2]. The explicit incidence list reproduced by Miller et al. [7, Fig. 8] permits the following reconstruction: delete points $50,\dots,59$, use the complete classes through $50,\dots,57$, split each surviving design group into its lower and upper five points, and complete the partial class through 59 with

$$\{8,18,28,38,48\}, \qquad \{9,19,29,39,49\}.$$

This gives $W(10,5) \geq 10$; the resulting schedule is listed in appendix A.2.

5 Finite lower bounds

Table 2: Constructive lower bounds, selected published comparisons, and the elementary partner-count ceiling E. The gap is the difference between E and the lower bound obtained here. A dash indicates that no published comparison is given.

Parameter	Published lower bound	This paper	E	Gap	Method
$W(10,5)$	≥ 9 [8]	≥ 10	12	2	ITD hole completion
$W(20,7)$	≥ 14 [4]	≥ 15	23	8	cyclic development
$W(21,7)$	≥ 8 [4]	≥ 16	24	8	cyclic development + split
$W(28,7)$	≥ 8 [4]	≥ 21	32	11	cyclic development + split
$W(14,13)$	—	≥ 6	15	9	explicit schedule

Table 2 compares the five constructions with selected published lower bounds. These comparisons do not constitute an exhaustive literature survey.

For $W(10,5)$, Miller et al. [8] obtain nine rounds from Brouwer's incomplete transversal design. Eight are projected complete classes; the ninth is a group-filling round. Completing one of the two partial classes as in proposition 5, using the incidence list in Miller et al. [7, Fig. 8], gives a tenth round. The complete schedule has 50 players, 100 blocks, and 1,000 distinct co-membership pairs.

For $W(20,7)$, Harvey and Winterer [4, Table 2] explicitly report 14 rounds. The array D_{20} in appendix A.1 has 15 columns satisfying the pairwise row-difference condition over $\mathbb{Z}_{20}$ and develops into 300 blocks with 6,300 distinct co-membership pairs.

For $W(21,7)$ and $W(28,7)$, the published lower bounds used for comparison come from six $7 \times n$ MOLR in Harvey and Winterer [4], translated by the construction of Sharma and Das [10]. Including the initial round and the divisibility round gives eight rounds because 7 divides both 21 and 28. The arrays D_{21} and D_{28} have respectively 15 and 20 developed rounds. Partitioning each latent class into blocks of seven adds one round, yielding 16 and 21 rounds. The resulting schedules contain respectively 7,056 and 12,348 distinct co-membership pairs.

For $W(14,13)$, tabu/min-conflicts search extended a five-round schedule by representing allowed pairs as a compatibility graph and finding a balanced partition into 14 cliques of size 13 (equivalently, independent sets in the conflict graph). The complete six-round schedule in appendix A.2 has 182 players, 84 blocks, and 6,552 distinct co-membership pairs. No published comparison is given: the relevant entry in Harvey and Winterer [4, Table 1] was unreadable in the available copy.

Every displayed construction has maximum pair multiplicity one. The cyclic cases follow from proposition 4 after checking the finite row-difference lists; the other two cases can be checked directly from the appended schedules.

6 Verification of the finite constructions

For each of the arrays D_{20}, D_{21}, and D_{28} in appendix A.1, subtract each pair of rows column by column modulo m. No difference repeats. Proposition 4 therefore shows that the columns develop into valid rounds. For $m = 21$ and $m = 28$, the additional latent-class partition described after that proposition gives the final round.

The complete schedules for $W(10, 5) \geq 10$ and $W(14, 13) \geq 6$ appear in appendix A.2. Each row of the tables is one group; the round labels identify the required partitions. Direct verification consists of checking that each round covers every player exactly once and that no unordered pair occurs in two groups.

The Python implementation in appendix B automates these checks. Player (i, x) in a cyclic construction is encoded by the integer $im + x$. The accompanying supplement contains the directory `anc/`, which supplies the five machine-readable inputs and the standalone validator `anc/validate_publication_claims.py`. Run it inside `anc/` as follows:

```
python3 validate_publication_claims.py all certificates
```

The validator expands and checks the supplied certificates; it does not replay the searches that discovered them.

Model and harness provenance

The computational research used ChatGPT under a Pro subscription and the Pi coding-agent harness connected to the OpenAI Codex provider. A ChatGPT Pro run with `gpt-5-6-pro` returned the concrete 9-5-10 schedule that motivated theorem 1; a Pi session with OpenAI Codex `gpt-5.6-sol` extracted and implemented the general class pattern. Pi with OpenAI Codex `gpt-5.5` developed and ran the randomized backtracking and row-compatibility search producing D_{20}, D_{21}, and D_{28}. ChatGPT Pro `gpt-5-6-pro` derived the tenth 10-5 round from the reproduced Brouwer ITD, and ChatGPT Pro with Extended Reasoning `gpt-5-5-pro` found the sixth 14-13 round by tabu/min-conflicts search.

These systems assisted construction discovery, generalization, coding, literature searching, and drafting. The author chose the questions and claims and takes responsibility for the manuscript.

7 Conclusion

Parallel-class splitting converts any suitable RTD$(p, 2p - 1)$ into one additional Social Golfer round. The construction applies whenever the required RTD exists. Given an RTD$(p, 2p)$, the analogous replacement works when the number of groups is $g = 2p$, but the resulting lower bound is already a known consequence of standard group filling. A related hole-completion argument turns Brouwer's incomplete transversal design into a ten-round schedule for 50 players in groups of five.

Cyclic difference arrays and two complete schedules establish four further constructive lower bounds, including $W(28, 7) \geq 21$ after a standard latent-group split. The cyclic arrays determine three schedules by translation; the other two schedules are listed explicitly.

Natural next questions include whether the class-splitting construction extends to other relations between g and p, whether more than one RTD class can be replaced compatibly, and whether the leaves of cyclic difference arrays admit additional factors beyond the standard latent-group split.

A Finite construction data

A.1 Cyclic arrays

Rows are indexed by i and columns by t. Apply equation (2) to obtain the rounds. For D_{21} and D_{28}, append the latent-class partition described in section 4.

D_{20} over $\mathbb{Z}_{20}$ (15 columns)

$i\backslash t$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	6	9	3	10	12	5	18	4	7	1	8	16	11	2
2	0	11	5	13	3	14	8	19	10	16	12	2	4	15	9
3	0	15	16	8	1	4	2	6	19	9	11	7	12	17	5
4	0	2	18	11	17	10	15	1	9	8	16	19	13	5	14
5	0	14	19	4	8	6	10	15	16	13	5	17	3	2	18
6	0	10	7	16	13	1	12	8	15	2	9	14	17	3	4

D_{21} over $\mathbb{Z}_{21}$ (15 columns)

$i\backslash t$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	7	11	6	18	2	5	8	4	14	20	9	15	10
2	0	2	15	8	12	16	6	10	17	3	7	11	1	9	5
3	0	7	19	18	9	17	13	1	10	12	15	5	3	4	8
4	0	13	12	14	8	3	11	19	18	16	9	10	5	7	17
5	0	20	9	16	3	14	12	11	1	5	18	6	8	13	4
6	0	11	5	3	10	12	18	14	19	6	17	16	2	1	20

D_{28} over $\mathbb{Z}_{28}$ (20 columns)

$i\backslash t$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	18	17	13	9	1	4	14	12	22	25	21	5	8	26	6	15	16	10	7
2	0	8	9	22	16	14	2	7	11	25	13	3	19	4	10	23	21	27	1	12
3	0	20	4	3	26	15	10	23	17	1	16	14	21	19	8	9	11	24	22	6
4	0	3	22	24	7	4	16	18	8	13	17	2	20	26	6	1	14	9	27	21
5	0	14	2	8	11	17	18	19	3	12	26	5	16	23	20	27	4	25	9	13
6	0	24	13	6	10	5	12	16	22	20	2	11	17	25	23	19	18	4	21	26

A.2 Complete schedules

Players are numbered from 0 to $gp - 1$. Each numbered collection of groups is one round.

A.2.1 A ten-round schedule for $W(10, 5)$

Round 1

1 $\{0, 15, 20, 33, 49\}$
2 $\{1, 10, 29, 34, 46\}$
3 $\{2, 16, 24, 38, 43\}$
4 $\{3, 18, 22, 30, 47\}$
5 $\{4, 19, 25, 32, 44\}$
6 $\{5, 13, 27, 31, 48\}$
7 $\{6, 11, 21, 39, 41\}$
8 $\{7, 14, 28, 35, 45\}$
9 $\{8, 12, 23, 37, 40\}$
10 $\{9, 17, 26, 36, 42\}$

Round 2

1 $\{0, 18, 23, 31, 44\}$
2 $\{1, 16, 21, 30, 49\}$
3 $\{2, 11, 29, 35, 47\}$
4 $\{3, 17, 25, 38, 40\}$
5 $\{4, 15, 28, 36, 46\}$
6 $\{5, 19, 26, 33, 45\}$
7 $\{6, 10, 24, 32, 48\}$
8 $\{7, 12, 22, 39, 42\}$
9 $\{8, 13, 20, 34, 41\}$
10 $\{9, 14, 27, 37, 43\}$

Round 3

1 $\{0, 14, 26, 38, 41\}$
2 $\{1, 18, 20, 32, 45\}$
3 $\{2, 17, 22, 31, 49\}$
4 $\{3, 12, 29, 36, 44\}$
5 $\{4, 13, 23, 39, 43\}$
6 $\{5, 16, 28, 37, 47\}$
7 $\{6, 19, 27, 30, 46\}$
8 $\{7, 11, 25, 33, 48\}$
9 $\{8, 10, 21, 35, 42\}$
10 $\{9, 15, 24, 34, 40\}$

Round 4

1 $\{0, 13, 29, 37, 45\}$
2 $\{1, 15, 27, 38, 42\}$
3 $\{2, 18, 21, 33, 46\}$
4 $\{3, 14, 23, 32, 49\}$
5 $\{4, 12, 26, 30, 48\}$
6 $\{5, 10, 20, 39, 40\}$
7 $\{6, 17, 28, 34, 44\}$
8 $\{7, 19, 24, 31, 47\}$
9 $\{8, 11, 22, 36, 43\}$
10 $\{9, 16, 25, 35, 41\}$

Round 5

1 $\{0, 19, 21, 36, 40\}$
2 $\{1, 17, 23, 35, 48\}$
3 $\{2, 15, 25, 39, 45\}$
4 $\{3, 10, 28, 31, 41\}$
5 $\{4, 11, 24, 37, 49\}$
6 $\{5, 14, 29, 30, 42\}$
7 $\{6, 12, 20, 38, 47\}$
8 $\{7, 18, 26, 34, 43\}$
9 $\{8, 16, 27, 33, 44\}$
10 $\{9, 13, 22, 32, 46\}$

Round 6

1 $\{0, 11, 28, 32, 42\}$
2 $\{1, 19, 22, 37, 41\}$
3 $\{2, 14, 20, 36, 48\}$
4 $\{3, 16, 26, 39, 46\}$
5 $\{4, 18, 27, 35, 40\}$
6 $\{5, 12, 25, 34, 49\}$
7 $\{6, 15, 29, 31, 43\}$
8 $\{7, 13, 21, 38, 44\}$
9 $\{8, 17, 24, 30, 45\}$
10 $\{9, 10, 23, 33, 47\}$

Round 7

1 $\{0, 17, 27, 39, 47\}$
2 $\{1, 12, 28, 33, 43\}$
3 $\{2, 19, 23, 34, 42\}$
4 $\{3, 15, 21, 37, 48\}$
5 $\{4, 10, 22, 38, 45\}$
6 $\{5, 18, 24, 36, 41\}$
7 $\{6, 13, 26, 35, 49\}$
8 $\{7, 16, 29, 32, 40\}$
9 $\{8, 14, 25, 31, 46\}$
10 $\{9, 11, 20, 30, 44\}$

Round 8

1 $\{0, 16, 22, 34, 48\}$
2 $\{1, 14, 24, 39, 44\}$
3 $\{2, 13, 28, 30, 40\}$
4 $\{3, 19, 20, 35, 43\}$
5 $\{4, 17, 29, 33, 41\}$
6 $\{5, 11, 23, 38, 46\}$
7 $\{6, 18, 25, 37, 42\}$
8 $\{7, 10, 27, 36, 49\}$
9 $\{8, 15, 26, 32, 47\}$
10 $\{9, 12, 21, 31, 45\}$

Round 9

1 $\{0, 1, 2, 3, 4\}$
2 $\{5, 6, 7, 8, 9\}$
3 $\{10, 11, 12, 13, 14\}$
4 $\{15, 16, 17, 18, 19\}$
5 $\{20, 21, 22, 23, 24\}$
6 $\{25, 26, 27, 28, 29\}$
7 $\{30, 31, 32, 33, 34\}$
8 $\{35, 36, 37, 38, 39\}$
9 $\{40, 41, 42, 43, 44\}$
10 $\{45, 46, 47, 48, 49\}$

Round 10

1 $\{0, 10, 25, 30, 43\}$
2 $\{1, 11, 26, 31, 40\}$
3 $\{2, 12, 27, 32, 41\}$
4 $\{3, 13, 24, 33, 42\}$
5 $\{4, 14, 21, 34, 47\}$
6 $\{5, 15, 22, 35, 44\}$
7 $\{6, 16, 23, 36, 45\}$
8 $\{7, 17, 20, 37, 46\}$
9 $\{8, 18, 28, 38, 48\}$
10 $\{9, 19, 29, 39, 49\}$

A.2.2 A six-round schedule for $W(14, 13)$

Round 1

1 {0, 14, 28, 42, 56, 70, 84, 98, 112, 126, 140, 154, 168}
2 {1, 15, 29, 43, 57, 71, 85, 99, 113, 127, 141, 155, 169}
3 {2, 16, 30, 44, 58, 72, 86, 100, 114, 128, 142, 156, 170}
4 {3, 17, 31, 45, 59, 73, 87, 101, 115, 129, 143, 157, 171}
5 {4, 18, 32, 46, 60, 74, 88, 102, 116, 130, 144, 158, 172}
6 {5, 19, 33, 47, 61, 75, 89, 103, 117, 131, 145, 159, 173}
7 {6, 20, 34, 48, 62, 76, 90, 104, 118, 132, 146, 160, 174}
8 {7, 21, 35, 49, 63, 77, 91, 105, 119, 133, 147, 161, 175}
9 {8, 22, 36, 50, 64, 78, 92, 106, 120, 134, 148, 162, 176}
10 {9, 23, 37, 51, 65, 79, 93, 107, 121, 135, 149, 163, 177}
11 {10, 24, 38, 52, 66, 80, 94, 108, 122, 136, 150, 164, 178}
12 {11, 25, 39, 53, 67, 81, 95, 109, 123, 137, 151, 165, 179}
13 {12, 26, 40, 54, 68, 82, 96, 110, 124, 138, 152, 166, 180}
14 {13, 27, 41, 55, 69, 83, 97, 111, 125, 139, 153, 167, 181}

Round 2

1 {8, 25, 35, 54, 58, 76, 85, 103, 115, 136, 144, 154, 177}
2 {0, 18, 34, 49, 59, 81, 97, 110, 122, 134, 142, 155, 173}
3 {7, 15, 33, 42, 64, 74, 96, 108, 125, 137, 149, 157, 170}
4 {13, 22, 30, 48, 57, 79, 89, 98, 123, 133, 152, 164, 172}
5 {6, 21, 37, 45, 56, 72, 94, 102, 113, 138, 148, 167, 179}
6 {11, 14, 36, 52, 60, 71, 87, 104, 117, 128, 153, 163, 175}
7 {4, 26, 29, 51, 67, 75, 86, 111, 112, 132, 143, 161, 178}
8 {3, 19, 41, 44, 66, 82, 90, 107, 119, 127, 140, 158, 176}
9 {2, 23, 40, 50, 69, 73, 84, 105, 118, 130, 151, 159, 169}
10 {1, 17, 38, 55, 65, 77, 88, 109, 120, 126, 145, 166, 174}
11 {5, 16, 32, 53, 63, 80, 92, 101, 124, 135, 141, 160, 181}
12 {9, 20, 31, 47, 68, 78, 95, 99, 116, 139, 150, 156, 168}
13 {12, 24, 28, 46, 62, 83, 93, 106, 114, 131, 147, 165, 171}
14 {10, 27, 39, 43, 61, 70, 91, 100, 121, 129, 146, 162, 180}

Round 3

1 {11, 20, 32, 50, 66, 77, 89, 100, 112, 138, 149, 155, 171}
2 {0, 25, 38, 46, 57, 82, 86, 101, 121, 139, 148, 159, 175}
3 {3, 15, 40, 53, 61, 72, 97, 98, 116, 136, 147, 163, 174}
4 {7, 18, 30, 55, 68, 76, 87, 107, 113, 131, 151, 162, 178}
5 {4, 22, 33, 45, 63, 83, 84, 99, 122, 128, 146, 166, 177}
6 {1, 19, 37, 48, 60, 78, 91, 110, 114, 137, 143, 154, 181}
7 {8, 16, 34, 52, 56, 75, 93, 109, 125, 129, 152, 158, 169}
8 {10, 23, 31, 42, 67, 71, 90, 106, 124, 133, 144, 167, 173}
9 {6, 26, 28, 47, 65, 81, 92, 105, 115, 127, 153, 164, 170}
10 {9, 14, 41, 43, 62, 80, 96, 103, 120, 130, 142, 161, 179}
11 {13, 24, 29, 49, 58, 70, 95, 102, 118, 135, 145, 157, 176}
12 {12, 21, 39, 44, 64, 73, 85, 111, 117, 126, 150, 160, 172}
13 {2, 27, 36, 54, 59, 79, 88, 108, 119, 132, 141, 165, 168}
14 {5, 17, 35, 51, 69, 74, 94, 104, 123, 134, 140, 156, 180}

Round 4

1 {13, 25, 37, 43, 59, 77, 84, 104, 124, 131, 150, 158, 170}
2 {0, 23, 32, 48, 64, 72, 95, 103, 119, 129, 153, 166, 178}
3 {1, 21, 40, 52, 58, 74, 92, 107, 112, 139, 146, 165, 173}
4 {2, 18, 41, 42, 65, 75, 91, 99, 123, 136, 148, 160, 171}
5 {3, 26, 34, 50, 60, 83, 85, 108, 121, 133, 145, 156, 179}
6 {4, 27, 30, 53, 62, 71, 94, 110, 115, 126, 149, 159, 176}
7 {5, 14, 39, 46, 66, 78, 97, 105, 113, 132, 152, 157, 177}
8 {6, 24, 31, 51, 63, 82, 88, 98, 117, 137, 142, 162, 169}
9 {7, 20, 29, 54, 61, 81, 93, 101, 120, 128, 140, 167, 172}
10 {8, 17, 33, 49, 57, 80, 90, 111, 114, 138, 151, 163, 168}
11 {9, 16, 36, 55, 67, 73, 89, 102, 122, 127, 147, 154, 180}
12 {10, 22, 28, 44, 69, 76, 96, 109, 116, 135, 143, 155, 175}
13 {11, 19, 35, 45, 68, 70, 86, 106, 125, 130, 141, 164, 174}
14 {12, 15, 38, 47, 56, 79, 87, 100, 118, 134, 144, 161, 181}

Round 5

1 {13, 20, 39, 42, 60, 80, 86, 107, 115, 134, 147, 166, 169}
2 {0, 27, 33, 51, 58, 77, 87, 106, 116, 127, 152, 160, 179}
3 {1, 22, 41, 53, 59, 70, 93, 105, 117, 138, 144, 156, 178}
4 {2, 25, 31, 52, 64, 83, 89, 110, 113, 135, 140, 161, 174}
5 {3, 21, 28, 54, 57, 75, 95, 104, 122, 130, 149, 162, 181}
6 {4, 15, 35, 48, 66, 73, 92, 109, 121, 131, 142, 167, 168}
7 {5, 24, 34, 45, 65, 71, 96, 100, 119, 139, 151, 154, 172}
8 {6, 17, 40, 46, 67, 79, 91, 103, 125, 128, 150, 155, 176}
9 {7, 26, 38, 44, 62, 78, 84, 102, 123, 129, 141, 163, 173}
10 {8, 14, 32, 55, 61, 82, 94, 99, 118, 133, 143, 165, 170}
11 {9, 19, 30, 50, 56, 81, 88, 111, 124, 136, 146, 157, 175}
12 {10, 16, 37, 49, 68, 74, 85, 98, 120, 132, 153, 159, 171}
13 {11, 23, 29, 47, 63, 76, 97, 108, 114, 126, 148, 158, 180}
14 {12, 18, 36, 43, 69, 72, 90, 101, 112, 137, 145, 164, 177}

Round 6

1 {47, 54, 42, 45, 51, 49, 44, 46, 43, 50, 53, 55, 48}
2 {120, 117, 124, 119, 116, 121, 118, 112, 113, 123, 125, 114, 122}
3 {146, 140, 151, 142, 152, 153, 149, 143, 150, 141, 148, 144, 147}
4 {135, 138, 136, 137, 128, 131, 129, 127, 134, 133, 139, 132, 126}
5 {38, 41, 36, 31, 32, 40, 33, 29, 28, 34, 35, 30, 39}
6 {78, 80, 71, 77, 72, 83, 73, 79, 76, 81, 75, 74, 70}
7 {24, 26, 14, 19, 25, 21, 23, 27, 17, 20, 15, 16, 18}
8 {178, 181, 174, 175, 170, 171, 177, 179, 176, 173, 172, 180, 169}
9 {37, 82, 22, 100, 97, 145, 67, 115, 168, 7, 160, 130, 52}
10 {94, 91, 85, 95, 88, 93, 96, 87, 84, 92, 89, 86, 90}
11 {58, 61, 60, 66, 62, 59, 63, 69, 57, 65, 56, 64, 68}
12 {110, 105, 98, 101, 102, 99, 106, 104, 109, 111, 108, 107, 103}
13 {9, 11, 5, 6, 8, 4, 0, 13, 3, 10, 2, 12, 1}
14 {157, 159, 156, 161, 162, 164, 163, 155, 166, 165, 154, 158, 167}

B A short verification program

The following program develops the three cyclic arrays and checks their schedules. The function `split_round` supplies the extra round when k divides m.

```
def parse_rows(text):
    return [[int(x) for x in row.split()]
            for row in text.strip().splitlines()]

D20 = parse_rows("""
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 6 9 3 10 12 5 18 4 7 1 8 16 11 2
0 11 5 13 3 14 8 19 10 16 12 2 4 15 9
0 15 16 8 1 4 2 6 19 9 11 7 12 17 5
0 2 18 11 17 10 15 1 9 8 16 19 13 5 14
0 14 19 4 8 6 10 15 16 13 5 17 3 2 18
0 10 7 16 13 1 12 8 15 2 9 14 17 3 4
""")

D21 = parse_rows("""
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 12 7 11 6 18 2 5 8 4 14 20 9 15 10
0 2 15 8 12 16 6 10 17 3 7 11 1 9 5
0 7 19 18 9 17 13 1 10 12 15 5 3 4 8
0 13 12 14 8 3 11 19 18 16 9 10 5 7 17
0 20 9 16 3 14 12 11 1 5 18 6 8 13 4
0 11 5 3 10 12 18 14 19 6 17 16 2 1 20
""")

D28 = parse_rows("""
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 18 17 13 9 1 4 14 12 22 25 21 5 8 26 6 15 16 10 7
0 8 9 22 16 14 2 7 11 25 13 3 19 4 10 23 21 27 1 12
0 20 4 3 26 15 10 23 17 1 16 14 21 19 8 9 11 24 22 6
0 3 22 24 7 4 16 18 8 13 17 2 20 26 6 1 14 9 27 21
0 14 2 8 11 17 18 19 3 12 26 5 16 23 20 27 4 25 9 13
0 24 13 6 10 5 12 16 22 20 2 11 17 25 23 19 18 4 21 26
""")
```

```
from itertools import combinations


def develop(D, m):
    """Develop a k-by-n array into n rounds on k*m players."""
    k, n = len(D), len(D[0])
    return [
        [
            [i * m + (a + D[i][t]) % m for i in range(k)]
            for a in range(m)
        ]
        for t in range(n)
    ]


def split_round(m, k):
    """Partition each latent class into consecutive k-subsets."""
    assert m % k == 0
    return [
        [i * m + q * k + r for r in range(k)]
        for i in range(k)
        for q in range(m // k)
    ]


def valid(rounds, g, p):
    """Check partitions and forbid repeated unordered pairs."""
    players = set(range(g * p))
    seen = set()
    for current in rounds:
        assert len(current) == g
        assert all(len(group) == p for group in current)
        assert set().union(*map(set, current)) == players
        assert sum(map(len, current)) == g * p
        for group in current:
            for pair in combinations(sorted(group), 2):
                assert pair not in seen
                seen.add(pair)
    return True


assert valid(develop(D20, 20), 20, 7)
assert valid(develop(D21, 21) + [split_round(21, 7)], 21, 7)
assert valid(develop(D28, 28) + [split_round(28, 7)], 28, 7)
```

For either explicitly listed schedule, transcribing the rounds in sections A.2.1 and A.2.2 as nested lists and passing them to `valid` performs the same direct check. The machine-readable schedules

and the validator for all five constructions are described in section 6.

References

[1] Robert F. Bailey and Andrea C. Burgess. Generalized packing designs. *Discrete Mathematics*, 313(11):1167–1190, 2013. doi: 10.1016/j.disc.2011.11.039.

[2] A. E. Brouwer. Four MOLS of order 10 with a hole of order 2. *Journal of Statistical Planning and Inference*, 10(2):203–205, 1984. doi: 10.1016/0378-3758(84)90072-7.

[3] Peter Danziger and Brett Stevens. Class-uniformly resolvable group divisible structures I: Resolvable group divisible designs. *The Electronic Journal of Combinatorics*, 11(1):R23, 2004. doi: 10.37236/1776.

[4] Warwick Harvey and Thorsten Jan Winterer. Solving the MOLR and social golfers problems. In *Principles and Practice of Constraint Programming—CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 286–300. Springer, 2005. doi: 10.1007/11564751_23.

[5] Dieter Jungnickel. On difference matrices, resolvable transversal designs and generalized Hadamard matrices. *Mathematische Zeitschrift*, 167(1):49–60, 1979. doi: 10.1007/BF01215243.

[6] Esther R. Lamken. Generalized balanced tournament designs. *Transactions of the American Mathematical Society*, 318(2):473–490, 1990. doi: 10.1090/S0002-9947-1990-0978380-6.

[7] Alice Miller, Ivaylo Valkov, and R. Julian R. Abel. Combinatorial solutions to the social golfer problem and the social golfer problem with adjacent group sizes. arXiv:2507.23376v2, 2025. Available at `https://arxiv.org/abs/2507.23376`; explicit incidence list in Figure 8.

[8] Alice Miller, Ivaylo Valkov, and R. Julian R. Abel. Combinatorial solutions to the social golfer problem and the social golfer problem with adjacent group sizes. *Symmetry*, 18(2):269, 2026. doi: 10.3390/sym18020269.

[9] Rolf S. Rees and Douglas R. Stinson. On resolvable group-divisible designs with block size 3. *Ars Combinatoria*, 23:107–120, 1987.

[10] V. K. Sharma and M. N. Das. On resolvable incomplete block designs. *Australian Journal of Statistics*, 27(3):298–302, 1985. doi: 10.1111/j.1467-842X.1985.tb00575.x.

[11] Markus Triska and Nysret Musliu. An improved SAT formulation for the social golfer problem. *Annals of Operations Research*, 194(1):427–438, 2012. doi: 10.1007/s10479-010-0702-5.

[12] Jianxing Yin. Cyclic difference packing and covering arrays. *Designs, Codes and Cryptography*, 37(2):281–292, 2005. doi: 10.1007/s10623-004-3991-3.