An Abstract Procedure to Compute Weak Schur Number Lower Bounds

Bruno Bouzy

LIPADE-TR-N° 2
September 2015

Technical Report
LIPADE
Laboratoire d'Informatique PAris DEscartes

An Abstract Procedure to Compute Weak Schur Number Lower Bounds

Bruno Bouzy[1]

[1] *LIPADE, Paris Descartes University, France*

Abstract

In this paper, we present a new method to compute explicit solutions to the weak Schur problem, and consequently obtain lower bounds to weak Schur numbers $WS(K)$ for $K \leq 8$. Our method is recursive, based on the principle that good solutions to the $K+1$ column problem are extensions of good solutions to the K column problem. We make several observations on the regularities of good solutions: (1) the last column is filled with nearly consecutive numbers. (2) The beginning of a column may contain holes. We test an abstract recursive function with refinements based on these observations. We define and implement an abstract simulation usable by Monte-Carlo search. Consequently, our algorithms search solutions in an abstract space whose size is several orders of magnitude smaller than the original space. Using such tools, we discovered new bounds: $WS(7) \geq 1736$ and $WS(8) \geq 5105$.

Keywords: *Weak Schur numbers, MAX Problems, Planning, Search*

1. Introduction

The Schur problem [1] consists in finding K partitions of the interval $[1, n]$ such that, in any partition, no number is the sum of any other numbers in that partition. It was proved that, for K partitions, and R operands of the sum, n has an upper bound, called the Schur number $S(K, R)$. When the sum is restricted to distinct numbers, this is

Email address: `bruno.bouzy@parisdescartes.fr` (Bruno Bouzy)

referred to as the Weak Schur problem, and the lowest upper bound is the Weak Schur number $WS(K,R)$. $WS(K,R)$ is the largest value of n that can be put into K partitions when considering sums of R distinct numbers. Lower bounds and upper bounds of the (Weak) Schur numbers have been proved [2], [3]. This problem is considered as an interesting problem in number theory [4] [5]. Very recently, some experimental studies have been launched to find the exact values or tighter bounds of Schur and Weak Schur numbers for low values of K, and for $R = 2$. It is easier to determine Weak Schur numbers rather than Schur numbers because of the additional constraint between the operands of the sum. In this paper, we are interested in finding Weak Schur numbers $WS(K)$ for $R = 2$ and low values of K, but as high as possible. In the state of the art, $WS(K)$ has been determined for $K = 1, 2, 3, 4$, and lower bounded for $K = 5, 6$, as shown in Table 1. Various state space search techniques have been applied: exhaustive tree search [6], tabu search [7], streamliners and constraint programming [8], and Monte-Carlo Search [9]. $S(K)$ has been determined for $K = 1, 2, 3, 4$, and lower bounded for $K = 5, 6, 7$, as shown in Table 2. The best lower bounds of $S(K)$ for $K = 6, 7$ are 536 and 1680 [10].

Table 1: Known values and bounds for Weak Schur Numbers.

K	1	2	3	4	5	6
$WS(K)$	2	8	23	66	≥ 196	≥ 582

Table 2: Known values and bounds for True Schur Numbers.

K	1	2	3	4	5	6	7
$S(K)$	1	4	13	44	≥ 160	≥ 536	≥ 1680

In this paper, we present a new method to compute lower bounds of $WS(K)$ based on an abstract and recursive procedure RWS. We observed regularities in the solutions built so far: the solutions contains groups of almost consecutive numbers. Therefore we present a method which builds abstract solutions containing groups of numbers. Our method searches for abstract solutions that stay in an abstract space. The size of the abstract space is many orders of magnitude smaller than the size of the actual space. For instance, for $K = 6$, which is the state-of-the-art number of columns so far, the size of the full space is 10^{450} that is a really huge number. We seek solutions in abstract spaces of size 10^{39} or even 10^9 only. The reduction of the size of the space in which solutions are searched is really huge. With our method, we confirm the values of the state of the art for $K \leq 6$ and provide new values for $K = 7$ and $K = 8$. The recursive and abstract procedure is translated into an abstract simulation, called *abstractSimulation*, to be used by Monte-Carlo Search (MCS), called *abstractMCS* in our work. RWS and *abstractMCS* find the state-of-the-art values for $K \leq 6$. Particularly, we confirm that $WS(6) \geq 582$. RWS and *abstractMCS* find new values for $K = 7$ and $K = 8$, showing that $WS(7) \geq 1736$ and $WS(8) \geq 5105$.

The first section of this paper defines the Weak Schur problem and describes the state of the art. The second section lists crucial observations that can be made regarding high quality solutions of the Weak Schur problem. Based on these observations, the third section presents the core idea of our method, *coreRWS*, and its refinement, RWS. The fourth section presents the translation of RWS into an abstract simulation, *abstractSimulation*, usable in *abstractMCS*. The fifth section presents the results of the experiments performed. The sixth section discusses the results. The last section concludes and lists possible future directions for research.

2. Weak Schur Numbers

2.1. Definition

Given a set of K columns, the problem of Weak Schur (WS) Numbers consists in placing a series of n consecutive integers in the set of columns such that n is as large as possible while respecting the following constraint: in a column, no integer can be the sum of two distinct integers in the same column. $WS(K)$ is the highest integer n such that the series $1, ..., n$ can be placed into a set of K columns. For instance, with $K = 1$, you may put 1 and 2 into the column but you cannot include 3 because $1 + 2 = 3$. Thus $WS(1) = 2$. For $K = 2$, you may put 1 and 2 into the first column and place 3 into the second one. Then you have the choice of placing 4 in column 1 or in column 2, and so on. $K = 2$ remains a simple puzzle. The placement of Table 3 works (where each column is shown as a row of the table). This solution shows that $WS(2) \geq 8$. No solution that includes 9 can be found for a set of 2 columns. Therefore $WS(2) = 8$.

Table 3: The optimal solution for $K = 2$.

1	2	4	8
3	5	6	7

More thinking for the reader is required to discover solutions for $K = 3$. However, this is still possible without the use of a computer. Table 4 shows the three solutions for $K = 3$.

Table 4: The three solutions A, B, C for $K = 3$. Obtained with a Depth-First Search.

A	1 2 4 8 11 22
	3 5 6 7 19 21 23
	9 10 12 13 14 15 **16** **17** 18 20
B	1 2 4 8 11 **17** 22
	3 5 6 7 19 21 23
	9 10 12 13 14 15 16 18 20
C	1 2 4 8 11 **16** 22
	3 5 6 7 19 21 23
	9 10 12 13 14 15 17 18 20

The reader may observe that all the three solutions for $K = 3$ contain the solution for $K = 2$ within their first two columns. Another observation is that the three solutions are very similar. The second one (solution B) differs from the first one (solution A) only by the position of 17 which is placed in column 1 instead of column 3. The third one (solution C) differs from solution A only by the position of 16 which is placed in column 1 instead of column 3. For $K \geq 4$, a computer is needed to find the solutions.

2.2. Related Work

A brute force method [6] showed that there are 29,931 length-66 solutions for $K = 4$ and no longer solution. Thus $WS(4) = 66$. In the appendix, table 14 shows a length-66 solution for $K = 4$. This solution for $K = 4$ contains solution C, which is optimal for $K = 3$. In 1952, Walker [5] claimed that $WS(5) = 196$, but he gave no explicit solution in his paper. Recently, several results were published. A tabu search method [7] showed that $WS(5) \geq 196$ and $WS(6) \geq 574$ by giving explicit solutions. Table 16 in Appendix B shows a length-196 solution for $K = 5$. This solution for $K = 5$ contains an optimal solution for $K = 4$. A study [11] claimed that $WS(6) \geq 575$ and gave a length-572 solution. A streamlined approach in a constraint programming framework [8] showed that $WS(6) \geq 581$ by giving an explicit solution. As far as we know, the most recent work is [9], who used a standard Monte-Carlo Search (MCS) approach [12] showing that $WS(6) \geq 582$ by giving the explicit solution of Table 17. This solution for $K = 6$ does not contain an optimal solution for $K = 5$, but contains a high quality solution (i.e., a length-195 solution). Table 1 sums up the results of this related work. Lower bounds have been proved by Bornzstein [13], and upper bounds by Abbott and Hanson [2], yielding the inequalities of (1).

$$(44/89)89^{K/4} \leq S(K) \leq WS(K) \leq \lfloor k!ke \rfloor \tag{1}$$

Finally, concerning true Schur numbers, it is worth noting inequality (2) [1]. Although it addresses true Schur numbers only, inequality (2) gives the ratio at which $WS(K)$ increases when K grows on average.

$$S(K+1) \geq 3S(K) + 1 \tag{2}$$

3. Observations

This section makes some observations concerning the kinds of searches that have been used to discover solutions. It defines the quality of a solution, and it discusses the structure of the best solutions found so far in the literature. Of course, most of these observations were already conjectured previously in the literature [7], [8], [9], but they were not used as explicitly as we do here.

3.1. Searches

On current computers, because the branching factor is 3, and the maximal depth is only 23, Depth-First Search (DFS) works almost instantly for $K = 3$. For $K = 4$, since the branching factor is 4, and the maximal depth is 66, DFS cannot complete the search in reasonable time. Standard Monte-Carlo Search (MCS) [12] works for $K = 6$ as shown by [9]. In this paper, standard MCS means that the simulations are performed by putting the numbers mainly one by one, and by ascending order. In contrast, *abstract* MCS means that an action consists in putting the numbers, group by group, still in ascending order, either in the simulations or in the tree browsed by MCS. The underlying idea of MCS is worth mentioning at this point. MCS is launched at a given level L and performs a level L simulation. To perform a level $L+1$ simulation, at each point of the $L+1$ simulation, for any legal action, MCS performs the action, and launches a level L simulation. MCS backs up the result with the action. Then it chooses the action with the best result, and plays it within the $L+1$ simulation. The best simulation played so far, is also memorized. The level 0 simulations of MCS are domain-dependent simulations. These level 0 simulations car be either standard [9] or abstract as shown by our work. It is worth mentioning that, in our work, the term abstract refers to the idea of putting numbers, group by group, and not to the MCS idea of nesting levels, which is another kind of abstraction. In MCS, the higher the level, the better the quality of the simulation.

3.2. Quality of a solution

For $K \leq 4$, the best solutions are proved to be optimal, and $WS(K)$ is known exactly. For $K = 5$, related work proved that 196 is a lower bound of $WS(K)$, but $WS(5) = 196$ is strongly conjectured. In this paper, we say that length-196 solutions are *experimentally optimal* for $K = 5$. We say that length-195 solutions are *high quality* solutions, or *good* solutions. The higher the length, the higher the solution quality. For $K \geq 6$, we know lower bounds only. For $K = 6$, 582 is the best lower bound known so far. In this paper, we say that length-582 solutions are experimentally optimal for $K = 6$. Of course, theoretically, we do not know wether length-582 solutions are optimal or not. Future work will say it. In this paper, we say that, for $K = 6$, the solutions of length not far from 582 are high quality solutions, or “good” solutions.

3.3. High quality solutions for K+1 columns include high quality solutions for K columns

Examining the solutions informs us that they are very highly structured. This was mentioned previously [8]. For instance, the three solutions for $K = 3$ contain the solution for $K = 2$ in their first two columns. Furthermore, among the $29,931$ solutions for $K = 4$, there are $8,238$ extensions of solution A for $K = 3$, and $21,693$ extensions of solution C. As shown in Table 16, the experimentally optimal solution for $K = 5$ contains an optimal solution for $K = 4$. This leads to the intuitive idea that experimentally optimal solutions for $K + 1$ columns include an experimentally optimal solution for K columns. However, this assertion seems untrue: see the solution for $K = 6$ of Table 17, which contains a length-195 solution for $K = 5$, and not a length-196 solution. If $WS(6) = 582$, this assertion is false. However, more carefully, we observe that high quality solutions, for $K + 1$ columns include a high quality solution for K columns.

3.4. Almost consecutive numbers in the last column are common

We observe that column $K + 1$ is always filled by a set of numbers that are almost consecutive. Let us elaborate on this observation. Let $First(K)$ be the first number put into column K, i.e., the smallest number in column K. The first sequence of consecutive numbers put into an empty column starts with $First(K)$ and stops with $2 \times First(K)$. Since $2 \times First(K) + 1 = First(K) + (First(K) + 1)$, $2 \times First(K) + 1$ cannot be put in the column. Thus, the maximal length of the sequence of consecutive numbers starting with $First(K)$ is $First(K) + 1$. Then, the length of the next longest sub-sequences of consecutive numbers that can be added in column K later on is at most $First(K)$. To illustrate this point, see that column 2 (respectively 3) of most high quality solutions contains many sub-sequences of length 3 (respectively 9) because $First(2) = 3$ (respectively $First(3) = 9$). As K is increasing, $First(K)$ increases as well. If the first K columns contain an optimal solution, then $First(K+1) = WS(K)+1$. If the first K columns do not contain an optimal solution, it is intuitively good to have $First(K)$ as high as possible, enabling longer sub-sequences to appear in column K later on.

3.5. Holes in the middle do not help

Let $[a, b]$ be a set of consecutive numbers in column K. ($b \leq 2a$). Then interval $[2a+1, 2b-1]$ is forbidden to appear in column K. For any m such that $a+2 < m < b-2$, if m is removed from column K, $[a, m-1] \cup [m+1, b]$ is in column K. Then the interval $[2a + 1, 2b - 1]$ is still forbidden to column K. This leads to the intuitive observation that there is no reason to remove a number such as m from an interval of consecutive numbers such as $[a, b]$. We refer to m as a *hole* of size one in the interval $[a, b]$; we now generalize to holes of larger sizes.

3.5.1. Definition

Let $[m, n]$ be an interval where $a < m \leq n < b$. Let us call $[m, n]$ a hole in $[a, b]$, where $s = 1 + n - m$ is the size of the hole.

3.5.2. Theorem

Creating a hole of size s in $[a, b]$ after the sequence of $c + 1$ consecutive numbers $[a, a + c]$, and before the sequence of $d + 1$ consecutive numbers $[b - d, b]$ gives the same set of forbidden numbers, i.e., $[2a + 1, 2b - 1]$, provided that $s < c$ and $s < d$.

3.5.3. Proof

The sets of forbidden numbers resulting from the combinations of intervals $[a, a + c]$ and $[b - d, b]$ are the union of $[2a+1, 2a+2c-1]$, $[a+b-d, a+c+b]$ and $[2b-2d+1, 2b-1]$. We have to prove that $2a + 2c - 1 \geq a + b - d - 1$ and that $a + c + b \geq 2b - 2d$, i.e. that $b - a <= 2c + d$ and that $b - a <= c + 2d$. This is true because $b - a = c + s + d + 1$ and $c > s$ and $d > s$.

To illustrate the fact that holes in the middle do not help, let us see the two solutions of tables 14 and 15. The first one has been obtained with a standard MCS. Let us consider column 4. Column 4 contains holes in the middle: 32, 38, 39 and 44. The set of forbidden numbers of column 4 is the interval $[50 - 97]$. The second solution is partly derived from the first one by moving numbers 32, 38, 39 and 44 to column 4. We observe that the set of forbidden numbers of column 4 remains the interval $[50 - 97]$. More specifically, the derivation between the two solutions has two steps. First, moving numbers into column 4. Secondly, moving numbers into column 3 (16 and 58 from column 1 to column 3). Although the two solutions have the same length, we see that the second solution has a clear upside over the first one. First, the sets of forbidden numbers of column 3 and 4 remains identical. Secondly, the sets of forbidden numbers of columns 1 and 2 are *strictly smaller* in the second solution than in the first solution. In the current work, we seek for solutions that have the structure of the solution of table 15.

3.6. Holes at the beginning of a column may help

Although the previous observation concerns the holes situated in the middle of a set of consecutive numbers, it is useful to have as many holes as possible at the *beginning* of a new sequence of almost consecutive numbers. This enables the creation of forbidden numbers as far as possible in the future, which makes the sequence as long as possible. Many columns of the solutions start with a hole after $First(K)$ or after $First(K) + N$. Therefore we defined some specific holes:

- $h0$: the absence of hole in the sequence,
- hN: a hole of size 1 at $First(K) + N$.
- hMN: the combination of hM and hN.

For instance $\{3, 5, 6, 7\}$ in column 2 is a $h1$. $\{9, 10, 12, 13\}$ in column 3 is a $h2$. With a short offline paper-and-pencil study, let us estimate the gain brought about by a given hole, hN or hMN in a given column K. Let us assume that the studied hole can be used in column K: the numbers corresponding to the hole are assumed to be put in the first $K - 1$ columns. Let NB be the gain of numbers brought about by applying a hole in column K. NB is the difference between the additional numbers that will be put at the end of the column and the size of the hole at the beginning of the column. Let END be the gain in the value of the ending number of the column. NB and END are properties to estimate the quality of the hole beforehand. Table 5 gives the values of NB and END for each kind of hole. $h0$ is the reference. $NB = 0$ for $h1$ means that we do not put the second number in the column, but we put an additional number at the end of the column. $END = 1$ for $h1$ means that the number at the end of the column equals the number at the end of the column if we were using $h0$, plus one. Concerning $h2$, $NB = 0$ means that we do not put the third number in the column, but we put an additional number at the end of the column. $END = 2$ for $h2$ means that the number put at the end of the column equals the number put at the end of the column if we were using $h0$, plus two. $h1$ and $h2$ are promising holes because one can expect to improve the value of the last number of the column. $h3$ and $h4$ are not promising because they lose one number in the column ($NB = -1$), and they do not increase the value of the last number of the column ($END = 0$). Holes of size 2 are promising because one can expect to improve the value of the last number of the column by 2, 3, or even 4. The greater the size of the hole, the better its possible consequences in terms of NB and END, but, unfortunately, the lower the chance that the numbers corresponding to the hole can be actually inserted into the previous columns.

Table 5: Gains brought about by holes (hN or hMN), in terms of NB (the number of numbers placed in the column) and END (the ending number of the column).

hN	$h0$	$h1$	$h2$	$h3$	$h4$
NB	0	0	0	-1	-1
END	0	1	2	0	0
hMN	$h12$	$h13$	$h14$	$h23$	$h34$
NB	0	0	0	0	-1
END	2	3	4	3	4

4. An Abstract and Recursive Procedure

This section describes an abstract and recursive procedure RWS to find solutions to the Weak Schur problem. First, it gives the basic idea with its core abstract procedure, $coreRWS$. Secondly, it describes the refinements to the procedure so as to actually find solution with high quality. From now on, the set of columns is named the *board*. In addition, a board B has a field $B.smallestOut$ which is the smallest number not put in a column of board B. In all the procedures, the numbers are put in ascending order. Therefore, $B.smallestOut$ is the next number to consider, and $B.smallestOut - 1$ is the number of consecutive numbers already put into board B.

4.1. The basic idea with its core procedure

We observed that the good solutions for K columns are made up of:

1. a high quality solution for $K - 1$ columns,
2. a sequence of almost consecutive numbers in column K, and
3. another solution for the first $K - 1$ columns.

```
$int\ coreRWS(B, K)$ ;
begin
    $If\ K > 1\ coreRWS(B, K-1)$ ;
    $fillColWithConsecutiveN(B, K)$ ;
    $If\ K > 1\ coreRWS(B, K-1)$ ;
    return $B.smallestOut - 1$ ;
end
```

Algorithm 1: The core RWS procedure.

The core of the procedure RWS follows this structure (Algorithm 1). First, starting with $B.smallestOut = 1$, *coreRWS* $(B, K-1)$ fills the first $K-1$ columns with a good solution. Secondly, starting with the new value of $B.smallestOut$, *fillCol With Consecutive* $N(B, K)$ fills column K with the longest sequence of consecutive numbers. Thirdly, starting with the new value of $B.smallestOut$, $coreRWS(B, K-1)$ fills the first $K-1$ columns with a good solution again. Finally, $B.smallestOut - 1$, the length of the solution, is returned.

Table 6 gives the solution output of $coreRWS(B, 2)$. For clarity, assuming that (1) a message is output at the beginning and at the end of each call of *coreRWS*, and (2) the numbers put into a column are also output, we give the output of *coreRWS* called for $K = 2$ in table 7. *coreRWS* fills 1 and 2 in the first column; 3 cannot be put into column 1 because $1 + 2 = 3$. 3, 4, 5, 6 are put in the second column; 7 cannot be put in column 2 because $3 + 4 = 7$; 7 is put in the first column.

Table 8 gives the solution output of $coreRWS(B, 3)$. For clarity, the output of *coreRWS* called for $K = 3$ is given by table 9. *coreRWS* fills the first two columns with the numbers from 1 up to 7 by the first call to $coreRWS(B, 2)$, then 8 to 16 are put in the third column. 17 cannot be put in column 3 because $8 + 9 = 17$. Then, the second call to $coreRWS(B, 2)$ start by considering 17. 17 is put in the first one, 18 up to 20 in the second one, 21 in the first one, and stops. 22 cannot be put in the first three columns because $21 + 1 = 22$, $19 + 3 = 22$, and $14 + 8 = 22$. The solutions obtained, if not optimal, are solutions with good quality: 7 instead of 8 for $K = 2$, and 21 instead of 23 for $K = 3$.

Table 6: A good solution for $K = 2$ obtained by $coreRWS(B, 2)$.

1-2 7
3-6

Table 7: Output of $coreRWS(B, 2)$.

```
coreRWS: begin, K = 2
   coreRWS: begin, K = 1
   FillCol 1: 1 2
   coreRWS: end, K = 1
FillCol 2: 3 4 5 6
   coreRWS: begin, K = 1
   FillCol 1: 7
   coreRWS: end, K = 1
coreRWS: end, K = 2
```

Table 10 gives the lengths of the sequences built by $coreRWS$ for $K = 1$ up to $K = 10$. As expected, given the simplicity of $coreRWS$, compared to the state of the art, the values are not positive contributions. For $K \leq 6$, the values are clearly inferior to the state of the art. Previous work on true Schur numbers experimentally proved that $S(7) \geq 1680$ [10]. Since inequalities (1) and (2) hold for any K, the values for $K \geq 7$ has a low quality. However, it is worth noting that $coreRWS$ is very simple and finds its results instantly on current computers. Next, we shall see how $coreRWS$ can be enhanced with important refinements.

4.2. Important refinements

Because the optimal solutions contain holes, and because they result from maximization, $coreRWS$ needs refinements. The first refinement consists in calling Depth-First-Search (DFS) when $K \leq 3$. This is possible because DFS performs its solution instantly when $K \leq 3$. This way, the placement of numbers in the first three columns will be always correct.

The second refinement consists in considering holes of given types when filling column

Table 8: A good solution for $K = 3$ obtained by $coreRWS(B, 3)$.

1-2 7 17 21
3-6 18-20
8-16

K. This refinement is important because it leads the search of abstract solutions into an adequate subspace of solutions structured as observed in the previous section. Let us call $SetOfConsideredHoles$ the set of holes considered. To fix ideas, $h0$, $h1$, $h2$, $h12$, $h13$, $h14$, and $h23$ are in $SetOfConsideredHoles$. This set can be widened to other holes or reduced. The third refinement consists in maximizing the length of the sequences over the possible holes. This gives Algorithm 2.

First, if K is less or equal than 3, this is the special case where DFS is called. In such case, $smallB$ is the sub-board of B made up with the first 3 columns of B. DFS is called on this board. Then $smallB$ is copied into the first 3 columns of B with $modifyFirst3ColOfWith(B, smallB)$, and RWS returns. In other cases, RWS calls itself on the first $K - 1$ columns. Then, $lmax$ and $bestB$ are declared to contain the current best information so far. $SetOfMatchingHoles$ is built with $buildSet$ $OfMatchingHoles$ (B). Among $SetOfConsideredHoles$, $buildSet$ $OfMatching$ $Holes$ (B) looks which holes could be actually put in the beginning of the sequence of consecutive numbers. To see this, for a given hole H, $buildSet$ $OfMatching$ $Holes$ (B) tests putting the numbers corresponding to H into the set of K-1 columns. If the test succeeds, H is inserted into $SetOfMatchingHoles$. $SetOfMatchingHoles$ being built, the loop is performed for each hole $Hole$ in $SetOfMatchingHoles$. $B2$ is a working board set to B at the beginning of each iteration of the loop. $fillCol$ $WithHole$ $(B2, K, Hole)$ fills column K according to the specification of $Hole$. $fillColWith$ $ConsecutiveN$ $(B2, K)$ remains identical as previously. $fillEndCol$ $(B2, K)$ fills column K so as to put the highest number as possible in it. To do this, creating holes in the end of column K is allowed provided that the numbers corresponding to these holes can be put in the first $K - 1$ columns of $B2$. $bestB$ and $lmax$ are updated when an improvement is seen. Before returning the length of the best sequence, $bestB$ is copied into B.

Table 9: Output of $coreRWS(B, 3)$.

```
coreRWS: begin, K = 3
   coreRWS: begin, K = 2
      coreRWS: begin, K = 1
      FillCol 1: 1 2
     coreRWS: end, K = 1
   FillCol 2: 3 4 5 6
      coreRWS: begin, K = 1
      FillCol 1: 7
      coreRWS: end, K = 1
   coreRWS: end, K = 2
FillCol 3: 8 9 10 11 12 13 14 15 16
   coreRWS: begin, K = 2
      coreRWS: begin, K = 1
      FillCol 1: 17
      coreRWS: end, K = 1
   FillCol 2: 18 19 20
      coreRWS: begin, K = 1
      FillCol 1: 21
      coreRWS: end, K = 1
   coreRWS: end, K = 2
coreRWS: end, K = 3
```

4.3. Other refinements

The first criterion to evaluate a K-column solution is its length: the longer, the better. However, since a K-column solution is intended to be used within a $(K+1)$-column solution, we want to distinguish two K-column solutions with a same length with another criterion that reflects its ability to be continued later on. This criterion must take into account the presence or absence of future forbidden numbers. When a K-column solution is completed, $B.smallestOut$ cannot be put in any of the K columns, so it will be put in column $K+1$. After this, the more allowed numbers in the first K columns, the better. Furthermore, the nearer the allowed numbers to $B.smallestOut$, the better. This process enables the creation of holes in column $K+1$. We define C_2 the second criterion, with Equation 3. However, in practice, the sum is finite. Actually,

Table 10: Values output by *coreRWS*.

K	1	2	3	4	5
$WS(K)$	2	7	21	61	180
K	6	7	8	9	10
$WS(K)$	536	1593	4733	13880	40844

five terms are sufficient.

$$C_2 = \sum_{n=1}^{\infty} A(n) 2^{-n} \tag{3}$$

$$A(n) = \begin{cases} 1 & \text{if } smallestOut + n \text{ is allowed} \\ 0 & \text{if } smallestOut + n \text{ is forbidden} \end{cases}$$

Since we want to use a Monte-Carlo approach in our experiments, we use randomness in the solution evaluations. We used a third criterion C_3 defined with Equation 4. Finally, the evaluation of a solution is given by Equation 5 with $\lambda \in [0, 1]$. E is called by DFS on terminal boards. The actual value of λ is set experimentally. $\lambda = 0.5$ works well.

$$C_3 = (rand()\%100)/100 \tag{4}$$

$$E = length + \lambda C_2 + (1 - \lambda) C_3 \tag{5}$$

For instance, for the 2-column solution of table 3, $smallestOut = 9$, 10, 12 are forbidden. 11, 13 and followers are allowed. Therefore, $C_2 = 0.25 + 0.125 = 0.375$ and $length + C_2 = 8.375$. For the 2-column solution of table 6, $smallestOut = 8$, 9 are forbidden. 10 and followers are allowed. Therefore, $C_2 = 0.5$ and $length + C_2 = 7.5$.

5. An abstract simulation for MC Search

As shown later on, the recursive procedure RWS produces very good results. However, it has a visible weakness. It locally maximizes the lengths of solutions within the call with $K - 1$ columns. It does not maximize globally over the whole process with K

```
$int\ RWS(B, K)$ ;
begin
    if $K \leq 3$ then
        $smallB = first3ColOf(B)$ ;
        $DFS(smallB)$ ;
        $modifyFirst3ColOfWith(B, smallB)$ ;
        return $B.smallestOut - 1$ ;
    end
    $RWS(B, K - 1)$ ;
    $lmax = 0$ ;
    $bestB = nil$ ;
    $SetOfMatchingHoles = buildSetOfMatchingHoles(B)$ ;
    for $Hole\ in\ SetOfMatchingHoles$ do
        $B2 = B$ ;
        $fillColWithHole(B2, K, Hole)$ ;
        $fillColWithConsecutiveN(B2, K)$ ;
        $fillEndCol(B2, K)$ ;
        $RWS(B2, K - 1)$ ;
        if $lmax < B2.smallestOut - 1$ then
            $lmax = B2.smallestOut - 1$ ;
            $bestB = B2$ ;
        end
    end
    $B = bestB$ ;
    return $lmax$ ;
end
```

Algorithm 2: The RWS procedure with its refinements.

columns. Therefore, we need to replace RWS by a new procedure that would maximize globally. Since MCS [12] was proved to work on the weak Schur problem [9], we want to use MCS with RWS as well. However, RWS cannot be used as such by MCS. MCS needs to call a simulation that can start at any point of the RWS procedure. We need to flatten RWS into a simulation usable by MCS. We call this procedure $abstractSimulation$. The term *abstract* still reflects that numbers are put, group by group, into the columns. $abstractSimulation$ can be called as a level 0 simulation in the framework of MCS. The top-level of MCS must be changed as well. At any step of a level L simulation, the set of holes matching the board must be computed. At any step, the set of actions corresponds to the set of holes matching the board. Abstract MCS mentions that the top-level of MCS is changed as above and uses $abstractSimulation$ as level 0 simulations. To this extent, the contribution of our work is to show that abstract MCS computes lower bounds for $K \leq 8$, while standard MCS does the same for $K \leq 6$ only.

We translated RWS into $abstractSimulation$. $abstractSimulation$ is a procedure iterating on stages. In $coreRWS$, a stage corresponds to a call to $fill\ ColWith\ ConsecutiveN$ $(Kloc)$. In table 9, there are 3 columns and 7 stages, each one corresponds to filling a specific column. On this example, the seven stages corresponds to filling columns 1, 2, 1, 3, 1, 2, 1 in this order. Processing $coreRWS$ for K columns corresponds to $2^K - 1$ stages. Let $stageN$ be the stage number. $stageN$ being given, it corresponds a unique column $Kloc = stageNToCol(stageN)$, where $stageNToCol\ (m)$ is the number of divisions of m to get an odd remainder. $stageNToCol(1) = 1$, $stageNToCol(2) = 2$, $stageNToCol(3) = 1$, $stageNToCol(4) = 3$, and so on. Therefore, if one wants to translate $coreRWS$ into a simulation, the simulation should be divided into $2^K - 1$ stages. In RWS, a stage corresponds to the calls to $fillCol\ WithHole$, $fillCol\ With\ ConsecutiveN$, and $fillEndCol\ (B2, K)$.

Similarly, processing RWS for K columns ($K \geq 3$) corresponds to $2^{K-2} - 1$ stages. Therefore, in order to translate $coreRWS$ into an abstract simulation, the simulation should be divided into $2^{K-2} - 1$ stages. Algorithm 3 shows the pseudo-code corresponding to an abstract simulation. $abstractSimulation$ can be used in MCS as a level 0 simulation. MCS can be called at any nesting level. The higher the level, the better the solutions, the longer the execution time.

First, $K \leq 3$ is a special case: DFS is used. Otherwise, the loop over the stages is entered. For each stage, the corresponding column $Kloc$ is determined. Then, for each iteration, if column $Kloc \leq 3$, then DFS is called to fill the first three columns.

```
$int\ abstractSimulation(B, K)$ ;
begin
    if $K \leq 3$ then
        $smallB = first3ColOf(B)$ ;
        $DFS(smallB)$ ;
        $modifyFirst3ColOfWith(B, smallB)$ ;
        return $B.smallestOut - 1$ ;
    end
    $NOfStages = 2^{K-2} - 1$ ;
    for  $stageN \leq NOfStages$ do
        $Kloc = 2 + stageNToCol(B, stageN)$ ;
        if $Kloc \leq 3$ then
            $smallB = first3ColOf(B)$ ;
            $DFS(smallB)$ ;
            $modifyFirst3ColOfWith(B, smallB)$ ;
        end
        else
            $Set = buildSetOfMatchingHoles(B)$ ;
            $Hole = holeRandomChoice(Set)$ ;
            $fillColWithHole(B, Kloc, Hole)$ ;
            $fillColWithConsecutiveN(B, Kloc)$ ;
            $fillEndCol(B, Kloc)$ ;
        end
        $stageN + +$ ;
    end
    return $B.smallestOut - 1$ ;
end
```

Algorithm 3: The abstract simulation for MC search.

Otherwise, column $Kloc$ is filled with the filling strategy of a hole drawn at random in Set, the set of possible holes matching board B. The process of matching holes with board B is identical to the one used in RWS. At the end, the length is returned. Then, $abstractSimulation$ can be used within MC search as a level 0 simulation.

6. Experiments

We used a 3.2 Ghz computer. For $K \leq 8$, memory use was not a concern, but the time used was. We set up $\lambda \in [0.5, 0.6]$. Table 11 gives the values found with RWS and $abstractMCS$ for $K \leq 8$. The quality of solutions found by $abstractMCS$ increases with the nesting level used. However, the time used by level $L+1$ is an order of magnitude greater than the time used by level L. The order of magnitude is linear in the branching factor, and linear in the depth of the search. We used a nesting level enabling $abstractMCS$ to find good solutions in few hours and not more. For $K \leq 6$, level 3 was sufficient to reproduce state-of-the-art results. Therefore, we think that level 3 is a good setting to launch $abstractMCS$ to find lower bounds which are tight enough. Table 12 gives a solution obtained for $K = 6$ with $abstractMCS$ at level 3 in 45 minutes. Table 18 gives a solution obtained for $K = 7$ with $abstractMCS$ at level 3 in 4 hours. Table 19 gives a solution obtained for $K = 8$ with $abstractMCS$ at level 2 in 6 hours. Processing $abstractMCS$ at level 3 for $K = 8$ could not produce any result before one day, and was interrupted before completion.

We think that the result obtained for $K = 7$ (1736) has a good quality. At least, it improves the lower bound brought about by the lower bound found for $K = 7$ on the true Schur problem (1680) [10]. 1736 is not far from $1746 = 3 \times 582$, 582 being the best lower bound found for $K = 6$. We think that the result obtained for $K = 8$ (5105) has an intermediate quality. On the one hand, it is obtained at level 2 only. On the other hand, 5105 is superior to $5041 = 3 \times 1680 + 1$, the lower bound brought about by [10] and equation 2. Given the limited quality obtained for $K = 8$ and the execution time used, we did not launch the experiment for $K = 9$.

Table 11: Best values found by RWS and $abstractMCS$.

K	1	2	3	4	5	6	7	8
$WS(K)$	2	8	23	66	196	582	1736	5105

Table 12: One solution for $K = 6$ obtained with an abstract Monte-Carlo Search.

1-2 4 8 11 16 22 25 53 63 68 136 149 154 177 182 192 197 394 407 412 435 440 450 455 521 526 531 536 541 564 569 **582**
3 5-7 19 21 23 50-52 64-66 137-139 150-152 179-181 193-195 395-397 408-410 437-439 451-453 523-525 537-539 566-568 579-581
9-10 12-15 17-18 20 54-62 140-148 183-191 398-406 441-449 527-530 532-535 570-578
24 26-49 153 155-176 178 411 413-434 436 540 542-563 565
67 69-135 454 456-520 522
196 198-393

We also observed the contribution of each kind of hole. $h0$ is always used by definition. $h1$ and $h2$ enable the algorithms to find $WS(6) \geq 582$. As expected, $h3$ and $h4$ do not help. $h12$, $h13$, $h14$ and $h23$ were included: they enhance the probability of finding $WS(6) \geq 582$. We did not use any other kind of hole in this set of experiments.

7. Discussion

An open question is whether an optimal solution for $K + 1$ columns is always an extension of an optimal solution for K columns. Since the current best solution for $K = 6$ is not an extension of a current best solution for $K = 5$, the answer may be negative. However, we think this is a false question, because it is more accurate to say that a high quality solution for $K+1$ columns is an extension of a high quality solution for K columns. This was the insight we followed to uncover RWS solutions. This property is correct for K less or equal than 8. We do not know if this property persists as K grows.

Our solutions are more compact than previous state-of-the-art solutions, particularly for columns with high K. To see this for $K = 6$, let us compare the *abstractMCS* solution of Table 12 with the MCS solution of Table 17. In column 1, there are fewer numbers. In column 2 of both solutions, the numbers are grouped 3 by 3, but the *abstractMCS* solution has fewer groups, and thus fewer numbers. Similarly for column 3 and 4. Conversely, in column 6, the *abstractMCS* solution has one long group of consecutive numbers (with one hole), and the MCS solution has many medium-size groups of consecutive numbers. Using the notation $a{-}b$, column 6 expression is far shorter in the *abstractMCS* solution than in the MCS solution. In column 5, the *abstractMCS* solution has two groups of consecutive numbers (with one hole in the first group and two holes in the second one), and the MCS solution has many groups of consecutive numbers with various sizes. Again, using the notation $a - b$, column 5 expression is far shorter in the *abstractMCS* solution than in the MCS solution. Finally, the reader may observe that the *abstractMCS* solution has 1 group in column K, 2 groups in column $K - 1$, and more generally 2^{K-k} groups in column k, i.e., approximately (without counting the times where column 1 is a relay of a hole in column K) 2^{K-1} numbers in column 1.

Why our method work for $K \leq 8$ while previous methods worked for $K \leq 6$ only? Our research uses an abstraction consisting in grouping numbers. This abstraction reduces the size of the space searched by several order of magnitudes. First, let us consider the size of the tree browsed by our algorithms with or without abstraction. With K columns and without abstraction, the depth of the tree roughly equals $WS(K)$, and the branching factor is K. An estimation of the size of the tree is $K^{WS(K)}$. With K columns and with abstraction, the depth of the tree equals $2^K - 1$, and the branching factor is $min(H, K)$, where H is the number of holes matching the board. An upper bound of H is 4. An estimation of the size of the tree with abstraction is H^{2^K-1}. Secondly, our algorithms use DFS when $K \leq 3$. With DFS used, the depth of the abstract tree equals $2^{K-2} - 1$ only. Table 13 yields the estimations of the tree sizes for $K \leq 8$. The numbers clearly show that using our abstraction and DFS greatly shortens the size of the searched space by several orders of magnitude, and enables our algorithms to work with $K = 7, 8$.

Table 13: Estimations of the size of the tree without abstraction (raw), with abstraction and without DFS (abs), and with abstraction and DFS (a+d).

	K	5	6	7	8
raw	$K^{WS(K)}$	10^{137}	10^{450}	10^{1400}	10^{4500}
abs	H^{2^K-1}	10^{19}	10^{39}	10^{79}	10^{159}
a+d	$H^{2^{K-2}-1}$	10^{5}	10^{9}	10^{19}	10^{39}

In our experiments, we used the set of holes $SetH = \{h0, h1, h2, h12, h13, h14, h23\}$.

To what extent the content of $SetH$ modifies the results obtained remains an open direction for future investigation.

Is 196 optimal for $K = 5$? Is 582 optimal for $K = 6$? These two questions remains theoretically open. These values remain lower bounds. However, the current work is yet another work that cannot surpass 196 and 582. The chance that these numbers are really the Weak Schur numbers for $K = 5$ and $K = 6$ increases. Actually, with $H \leq 4$, for $K = 5, 6$, the sizes of the searched spaces are small (10^5 and 10^9), and there is very few chances that our search missed the optimal solution if it has the structure we look for.

We discovered a first solution for $K = 7$ with length 1736, and a first one for $K = 8$ with length 5105, which constitute lower bounds to improve.

Is our method complete ? Our method is on a par with previous records for $K \leq 6$ and was the first method to find values for $K = 7$ and $K = 8$. Our method covers all previous results and extends them for $K = 7$ and $K = 8$. However, our method seeks for solutions in a sub-space of the solution space only. If the optimal solutions stay out of this sub-space, our method is not complete.

The process by which RWS fills the K columns looks like the complementary of the construction of the first K steps of the Cantor set. For $K = 1$, the set of numbers in column 1 corresponds to the $[0, 1]$ interval. For $K = 2$, the set of numbers in column 2 corresponds to the $[1/3, 2/3]$ interval, and the set of numbers in column 1 corresponds to the $[0, 1/3]$ and $[2/3, 1]$ intervals. For $K = 3$, the set of numbers in column 3 corresponds to the $[1/3, 2/3]$ interval, and the set of numbers in column 2 corresponds to the $[1/9, 2/9]$ and $[7/9, 8/9]$ intervals, and the set of numbers in column 1 corresponds to the $[0/9, 1/9]$, $[2/9, 3/9]$, $[6/9, 7/9]$ and $[8/9, 1]$ intervals. This pattern continues for higher values of K. With this correspondence, it is worth noting that a number situated in column k in a RWS solution would correspond to a number in $[0, 1]$ whose base 3 representation contains 0 or 2 only in the first $K - k$ digits and a 1 at the $K - k + 1$ digit.

Can our method be adapted for true Schur numbers? The answer is negative. It is known that $S(1) = 1$, $S(2) = 4$, $S(3) = 13$, and $S(4) = 44$ [14], [15]. In our experiments performed on true Schur numbers, we observed that RWS could find the correct $S(K)$ for $K \leq 3$ but found out 40 only for $K = 4$. Furthermore, the explicit sequences of length 44 found for $K = 4$ [14] and for $K = 6, 7$ [10] do not match the structure of sequences produced by RWS. Therefore, adapting our method to true Schur numbers remains as future work. To deal with true Schur numbers in a similar way as we did in the current work, we have to seek a structure in the solutions, and design a method that would produces solutions with such structure.

8. Conclusion

In this work, we presented a new method to compute lower bounds of Weak Schur numbers. It is an abstract and recursive method based on the principle that high quality solutions for $K+1$ columns are extensions of high quality solutions for K columns. It is based on two main observations. First, the last column can be filled with almost consecutive numbers, and making holes in the middle of a sequence of consecutive numbers does not help. Secondly, making holes at the beginning of a sequence is appropriate because it lengthens the end of the sequence. Our method computes abstract solutions. Rather than putting the numbers one by one into the column, our method puts groups of almost consecutive numbers in the columns. We explicited two recursive and abstract procedures to approximate $WS(K)$: $coreRSW$, the basic version, and RWS, the refined version. We also translated RWS into the procedure *abstractSimulation*, an abstract simulation which can be launched within Monte-Carlo Search. By doing so, the size of the studied space is smaller than the size of the whole space by several orders of magnitude. This size reduction enabled our method to deal with problems for $K = 7, 8$, where previous methods only dealt with problems for $K \leq 6$. Our method confirmed the previous lower bounds found for $K \leq 6$. Furthermore, our method found new lower bounds for $K = 7$ and $K = 8$. Our approach, either with RWS or with *abstractSimulation* within MCS, proves that $WS(6) \geq 582$, $WS(7) \geq 1736$ and $WS(8) \geq 5105$.

Next, several followings of this research are foreseeable. First, we aim to compute high quality lower bounds for $K \geq 9$. Secondly, we aim to adapt our method for true Schur numbers. Thirdly, we seek to discover better theoretical lower and upper bounds based on the current work.

References

[1] I. Schur, Uber die kongruenz $x^m + y^m = z^m (mod p)$, Jahresbericht des Deustchen Mathematiker Vereinigung 25 (1916) 114–117.

[2] H. Abbott, D. Hanson, A problem of Schur and its generalizations, Acta Arithmetica 20 (1972) 175–187.

[3] R. W. Irving, An extension of Schur's theorem on sum-free partitions, Acta Arithmetica.

[4] R. Rado, Some solved and unsolved problems in the theory of numbers, The Mathematical Gazette 25 (264) (1941) 72–77.

[5] G. Walker, A problem in partitioning, American Math. Monthly 59 (253).

[6] P. F. Blanchard, F. Harary, R. Reis, Partitions into sum-free sets, Integers A7 (6).

[7] D. Robilliard, C. Fonlupt, V. Marion-Poty, A. Boumaza, A multilevel tabu search with backtracking for exploring weak Schur numbers, in: Proceedings of Evolution Artificielle, 2011.

[8] R. L. Bras, C. P. Gomes, B. Selman, From streamlined combinatorial search to efficient constructive procedures, in: Proceedings of AAAI-12, 2012.

[9] S. Eliahou, C. Fonlupt, J. Fromentin, V. Marion-Poty, D. Robilliard, F. Teytaud, Investigating Monte-Carlo methods on the weak Schur problem, in: M. Middendorf, C. Blum (Eds.), Evolutionary Computation in Combinatorial Optimization, Vol. 7832 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 191–201.

[10] H. Fredricksen, M. M. Sweet, Symmetric sum-free partitions and lower bounds for schur numbers, Electronic Journal of Combinatorics 7 (1) (2000) 1–9.

[11] S. Eliahou, J. M. Marín, M. P. Revuelta, M. I. Sanz, Weak Schur numbers and the search for G. W. Walker's lost partitions, Computers and Mathematics with Applications 63 (1) (2012) 175–182.

[12] T. Cazenave, Nested Monte-Carlo Search, in: IJCAI, 2009.

[13] P. Bornzstein, On an extension of a theorem of Schur, Acta Arithmetica 101 (2002) 395–399.

[14] H. Abbott, L. Moser, Sum-free sets of integers, Acta Arithmetica 11 (1966) 393–396.

[15] L. Baumert, Sum-free sets, unpublished (1961).

Appendix

This section gathers solutions mentioned above.

Table 14: One solution for $K = 4$. (This solution was obtained with a normal Monte-Carlo Search, and not with our abstract Monte-Carlo Search.)

1 2 4 8 11 **16** 22 25 **32** **44** 53 **58** 63
3 5-7 19 21 23 **38** **39** 50-52 64-**66**
9 10 12-15 17 18 20 54-57 59-62
24 26-31 33-37 40-43 45-49

Table 15: Compact solution for $K = 4$, derived from solution of table 14. The derivation has two steps. First, moving numbers into column 4 without altering the set of forbidden numbers of column 4 (32 and 44 from column 1 to column 4, and 38-39 from column 2 to column 4). Secondly, moving numbers into column 3 without altering the set of forbidden numbers of column 3 (16 and 58 from column 1 to column 3).

1 2 4 8 11 22 25 53 63
3 5-7 19 21 23 50-52 64-**66**
9 10 **12-18** 20 **54-62**
24 **26-49**

Table 16: One solution for $K = 5$, obtained with normal Monte-Carlo Search.

1 2 4 8 11 16 22 25 31 45 50 60 63 69 106 135 140 150 155 178 183 **196**
3 5-7 19 21 23 35 51-53 64-66 77-79 137-139 151-153 180-182 193-195
9 10 12-15 17 18 20 54-59 61 62 99-105 141-149 184-192
24 26-30 32-34 36-44 46-49 98 154 156-177 179
67 68 70-76 80-97 107-134 136

Table 17: The solution for $K = 6$ [9].

1-2 4 8 11 22 25 40 50 63 68 73 82 87 92 97 116 121 133 139 149 154 159 177 182 187 192 197 252 304 342 370 394 407 412 417 435 440 445 450 455 464 469 474 479 488 493 502 507 521 526 536 541 554 564 569 **582**
3 5-7 19 21 23 37 51-53 64-66 79-81 93-95 109-111 122-124 136-138 150-152 167-168 179-181 193-195 368 395-397 408-410 424-425 437-439 451-453 465-467 480-482 495-497 512 523-525 537-539 551-553 566-568 579-581
9-10 12-18 20 54-62 103-108 140-148 183-186 188-191 398-406 441-444 446-449 486-487 490 492 494 527-535 570-578
24 26-36 38-39 41-49 98-102 153 155-158 160-166 169-176 178 292 411 413-416 418-423 426-434 436 540 542-550 555-563 565
67 69-72 74-78 83-86 88-91 96 112-115 117-120 125-132 134-135 454 456-463 468 470-473 475-478 483-485 489 491 498-501 503-506 508-511 513-520 522
196 198-251 253-291 293-303 305-341 343-367 369 371-393

Table 18: One solution for $K = 7$ obtained with an abstract Monte-Carlo Search.

1-2 4 8 11 16 22 25 50 63 68 136 149 154 177 182 187 192 197 394 397 407 412 435 440 450 455 521 526 531 536 541 564 569 582 1170 1175 1180 1185 1208 1213 1218 1223 1228 1294 1299 1309 1314 1337 1342 1355 1555 1565 1570 1593 1598 1603 1608 1613 1679 1684 1694 1699 1722 1727
3 5-7 19 21 23 51-53 64-66 137-139 150-152 179-181 193-195 395-396 408-410 437-439 451-453 523-525 537-539 566-568 579-581 1167-1169 1181-1183 1210-1212 1224-1226 1296-1298 1310-1312 1339-1341 1352-1354 1552-1554 1566-1568 1595-1597 1609-1611 1681-1683 1695-1697 1724-1726
9-10 12-15 17-18 20 54-62 140-148 183-186 188-191 398-406 441-449 527-530 532-535 570-578 1171-1174 1176-1179 1214-1217 1219-1222 1300-1308 1343-1351 1556-1564 1599-1602 1604-1607 1685-1693 1728-**1736**
24 26-49 153 155-176 178 411 413-434 436 540 542-563 565 1184 1186-1207 1209 1313 1315-1336 1338 1569 1571-1592 1594 1698 1700-1721 1723
67 69-135 454 456-520 522 1227 1229-1293 1295 1612 1614-1678 1680
196 198-393 1356-1551
583-1166

Table 19: One solution for $K = 8$ obtained with an abstract Monte-Carlo Search.

1-2 4 8 11 22 52 55 62 67 137 147 152 178 181 188 193 389 399 405 430 435 440 445 510 515 525 531 556 559 566 571 1145 1155 1161 1186 1189 1196 1201 1266 1271 1281 1286 1309 1315 1322 1327 1518 1523 1533 1561 1564 1567 1574 1579 1644 1649 1659 1687 1690 1700 1705 3410 3413 3423 3451 3454 3457 3464 3469 3534 3539 3549 3555 3580 3585 3590 3595 3786 3791 3801 3806 3832 3835 3842 3847 3912 3917 3927 3932 3955 3968 3973 4542 4547 4552 4557 4562 4588 4591 4598 4603 4668 4673 4683 4711 4724 4729 4920 4925 4935 4966 4976 4981 5046 5051 5061 5066 5089 5102
3 5-7 19 21 23 49-51 63-65 134-136 148-150 175-177 189-191 386-388 400-402 427-429 441-443 512-514 526-528 553-555 567-569 1142-1144 1156-1158 1183-1185 1197-1199 1268-1270 1282-1284 1310-1312 1323-1325 1520-1522 1534-1536 1562-1563 1575-1577 1646-1648 1660-1662 1688-1689 1701-1703 3411-3412 3424-3426 3452-3453 3465-3467 3536-3538 3550-3552 3577-3579 3591-3593 3788-3790 3802-3804 3829-3831 3843-3845 3914-3916 3928-3930 3956-3958 3969-3971 4544-4546 4558-4560 4585-4587 4599-4601 4670-4672 4684-4686 4712-4714 4725-4727 4922-4924 4936-4938 4963-4965 4977-4979 5048-5050 5062-5064 5090-5092 5103-**5105**
9-10 12-18 20 53-54 56-61 138-146 179-180 182-187 390-398 431-434 436-439 516-524 557-558 560-565 1146-1154 1187-1188 1190-1195 1272-1280 1313-1314 1316-1321 1524-1532 1565-1566 1568-1573 1650-1658 1691-1699 3414-3422 3455-3456 3458-3463 3540-3548 3581-3584 3586-3589 3792-3800 3833-3834 3836-3841 3918-3926 3959-3967 4548-4551 4553-4556 4589-4590 4592-4597 4674-4682 4715-4723 4926-4934 4967-4975 5052-5060 5093-5101
24-48 151 153-174 403-404 406-426 529-530 532-552 1159-1160 1162-1182 1285 1287-1308 1537-1560 1663-1686 3427-3450 3553-3554 3556-3576 3805 3807-3828 3931 3933-3954 4561 4563-4584 4687-4710 4939-4962 5065 5067-5088
66 68-133 444 446-509 511 1200 1202-1265 1267 1578 1580-1643 1645 3468 3470-3533 3535 3846 3848-3911 3913 4602 4604-4667 4669 4980 4982-5045 5047
192 194-385 1326 1328-1517 1519 3594 3596-3785 3787 4728 4730-4919 4921
570 572-1141 3972 3974-4541 4543
1704 1706-3409