Investigating Monte-Carlo Methods on the Weak Schur Problem

Shalom Eliahou[1], Cyril Fonlupt[2], Jean Fromentin[1], Virginie Marion-Poty[2], Denis Robilliard[2], and Fabien Teytaud[2]

[1] {eliahou, fromentin}@lmpa.univ-littoral.fr
Univ Lille Nord de France
ULCO, LMPA, BP 699
F-62228 Calais, France
[2] {fonlupt,poty,robilliard,teytaud}@lisic.univ-littoral.fr
Univ Lille Nord de France
ULCO, LISIC, BP 719
F-62228 Calais, France

Abstract. Nested Monte-Carlo Search (NMC) and Nested Rollout Policy Adaptation (NRPA) are Monte-Carlo tree search algorithms that have proved their efficiency at solving one-player game problems, such as morpion solitaire or sudoku 16x16, showing that these heuristics could potentially be applied to constraint problems. In the field of Ramsey theory, the *weak Schur number* $WS(k)$ is the largest integer n for which their exists a partition into k subsets of the integers $[1, n]$ such that there is no $x < y < z$ all in the same subset with $x + y = z$. Several studies have tackled the search for better lower bounds for the Weak Schur numbers $WS(k), k \geq 4$. In this paper we investigate this problem using NMC and NRPA, and obtain a new lower bound for $WS(6)$, namely $WS(6) \geq 582$.

1 Introduction

Nested Monte-Carlo Search (NMC) [5] and the recent Nested Rollout Policy Adaptation (NRPA) [17] are Monte-Carlo tree search algorithms that have proved their efficiency at solving constrained problems, although mainly in the AI field (e.g. sudoku 16x16, morpion solitaire game, Same Game). Within the field of Ramsey theory, challenging problems are the search for the Van der Waerden numbers [4] and for the Schur numbers [10], or the search for better lower or higher bounds for these numbers. Finding new lower bounds can be tackled with computational tools, by constructing a mathematical object that exhibits the required properties. In general this construction implies exploring a heavily constrained combinatorial space of huge dimension. In this paper we investigate the search for better lower bounds for the so-called Weak Schur numbers, using NMC and NRPA. First, we present the definition of Weak Schur numbers, next we recall the principles of the two Monte-Carlo search algorithms that we used, then we compare the results obtained, notably the discovery of a new lower bound $WS(6) \geq 582$, before concluding.

2 Weak Schur numbers

The Weak Schur numbers originate from two Ramsey theory theorems, dating back to the first half of the 20^{th} century. We recall their definition and the current state of knowledge on this topic, before presenting experimental data that motivate the choice of Monte-Carlo search methods.

2.1 Mathematical description

First, we explain the concept of Schur numbers. A set P of integers containing no elements x, y, z with $x + y = z$ is called *sum-free*. A theorem of Schur [18] states that, given $k \geq 1$, there is a largest integer n for which the integer interval set $[1, n]$ (i.e. $\{1, 2, \ldots, n\}$) admits a partition into k sum-free sets. The largest such integer n is called the k-th Schur number, and is denoted by $S(k)$.

As a somewhat weaker notion than sum-free, a set P of integers containing no *pairwise distinct* elements x, y, z with $x + y = z$ is called *weakly sum-free*. A result similar to that of Schur was shown by Rado [13] : given $k \geq 1$, there is a largest integer n for which the interval set $[1, n]$ admits a partition into k weakly sum-free sets. This largest such integer n is called the k-th Weak Schur number, and is denoted by $WS(k)$.

For example, in the case $k = 2$, a weakly sum-free partition of the first 8 integers is provided by:

$$\{1, 2, 3, 4, 5, 6, 7, 8\} = \{1, 2, 4, 8\} \cup \{3, 5, 6, 7\}.$$

It is straightforward to verify, with exhaustive search, that increasing the interval to $[1, 9]$ yields a set that does not admit any weakly sum-free partition into 2 sets, thus we have $WS(2) = 8$.

2.2 State of the art

The exact values of $WS(k)$ (and of $S(k)$) are only known up to $k = 4$:

– $WS(1) = 2$ and $WS(2) = 8$ are easily verified
– $WS(3) = 23$, $WS(4) = 66$ were shown by exhaustive computer search in [2].

For the general case $k \geq 5$, known results are a lower bound from Abbott and Hanson [1] and an upper bound by Bornsztein [3]:

$$c89^{k/4} \leq S(k) \leq WS(k) \leq \lfloor k!ke \rfloor$$

with c a small positive constant.

Some special cases are better known through experimental studies. In [2] it was shown that: $WS(5) \geq 189$, while a much older note by Walker [19] claimed, without proof, that $WS(5) = 196$. More recently, [8] provided a weakly sum-free partition of the set $[1, 196]$ in 5 sets, confirming Walker's claim that $WS(5)$ is at least as large as 196, and also gave a weakly sum-free partition of $[1, 572]$ in 6 sets, implying $WS(6) \geq 572$. In [9] the lower bound for $WS(6)$ was pushed to $WS(6) \geq 574$ using meta-heuristic search. This result was superseded by [12] establishing a current best bound at $WS(6) \geq 581$.

This result in [12] was obtained with a constraint solver. Extra constraints, nicknamed *streamliners*, were added to the problem in order to reduce the search space size, with the (unproven) assumption that optimal solutions were preserved. Some of these streamliners were taken either from the previous studies on this subject [8, 9] or from the literature on the related (non weak) Schur problem [10].

2.3 Methodology and experimental data

In this paper, we consider using AI methods, namely variants of Monte Carlo Tree Search (MCTS) in order to tackle the search of better lower bounds for $WS(5)$ and $WS(6)$. The motivation behind this method choice is supported by a study of weakly sum-free 3-partitions of $[1, WS(3)]$ and 4-partitions of $[1, WS(4)]$. We now explain this last case.

A weakly sum-free 4-partition (i.e. partition into 4 subsets) of $[1, n]$ can be coded by a word, $w = a_1a_2 \dots a_n$, where the letters $\{a_i\}$ are in the symbol set $\{1, 2, 3, 4\}$ and $a_i = j$ means that integer i is in the subset j. For example, the word $1, 1, 2, 1, 3, 4$ encodes the partition $\{\{1, 2, 4\}, \{3\}, \{5\}, \{6\}\}$. Up to symmetry, we can systematically enumerate in lexicographic order all words associated to partitions that are *terminal*, i.e. such that it is not possible to extend the partition by placing the next successive integer while keeping the weakly sum-free property. An example of such a word is given by $w =$ 112233444444442333332221, associated to the terminal partition below:

{
  {1, 2, 24},
  {3, 4, 15, 21, 22, 23},
  {5, 6, 16, 17, 18, 19, 20},
  {7, 8, 9, 10, 11, 12, 13, 14}
}

Clearly it is not possible to add 25 in any subset of this partition, thus it is terminal, and we also call w a terminal word. An exhaustive computer enumeration shows that there are exactly $536\,995\,391\,721$ terminal words that correspond to terminal 4-partitions. To reduce the size of the data, we cluster them in groups of 1 billion consecutive words in lexicographic order, and then plot the mean and maximal word length (including solutions of maximal length $WS(4) = 66$) from each group in Figure 1. Mean and maximal length exhibit a covariance correlation of 0.7, thus they are somewhat correlated, and the same behavior is also observed for 3-partitions. This means that good samples on average, can lead to better solutions[3]. Assuming that this good property also holds for 5-partitions and 6-partitions, Monte Carlo sampling appears as a promising method for searching lower bounds for $WS(5)$ and $WS(6)$. It is also noticeable that optimal solutions are clustered in the beginning with respect to the lexicographic ordering.

[3] Note that, quite interestingly, this is not the case for standard Schur Numbers, at least $S(3)$ and $S(4)$.

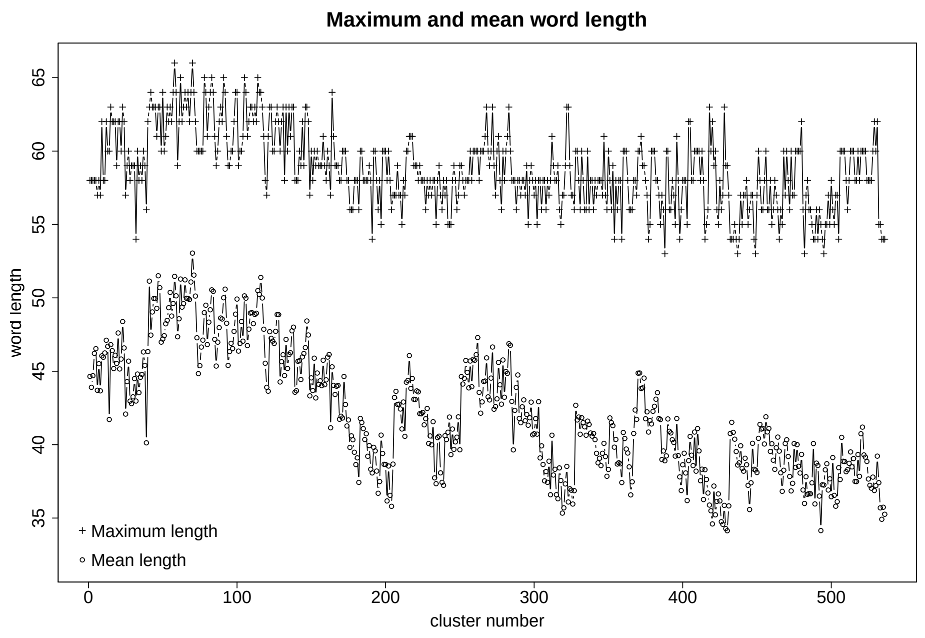

Fig. 1. Mean and maximal length of terminal word groups associated to 4-partitions, listed in lexicographic order.

3 Methods

In this section we present the two Monte-Carlo heuristics that were used for our experiments.

3.1 Nested Monte-Carlo Search

The Nested Monte-Carlo Search (NMC) algorithm [5] is a tree search algorithm. The basic idea is to incrementally build a solution with the particularity that each decision is based on the result of lower level calls of the algorithm. The lowest level, the level 0, is a single Monte-Carlo playout. A Monte-Carlo playout is a sequence for which each remaining decision is made randomly, until no more possible decision can be made. For higher levels > 0, all possible decisions are tried (i.e. recursively explored) and the branch of the search tree associated to the best result is chosen as the new decision.

The pseudo-code of NMC is presented in Algorithm 1, where:

- a $position$ describes the state of the solution which is being constructed (the root position is empty, no decision having been chosen already). Here a solution is a weakly sum-free k-partition. Notice that $position$ is always passed as argument by copy, and never by reference.
- `play`$(position, d)$ is a function that returns the new position obtained after having performed decision d relatively to $position$. In our Weak Schur numbers problem implementation, a decision consists in choosing in which subset of the partition to place the next integer. Notice that the selection of the next integer is deterministic (see Sect. 4.1) and is not part of the decision.
- `MonteCarlo`$(position)$ is a function that completes the $position$ by playing random decisions, until the k-partition is terminal (see Sect. 2.3). It returns a 2-tuple: the evaluation of the terminal partition, and the sequence of decisions that were made to obtain it, called playout in Algorithm 1. As integers are not always played in successive order, possibly leaving "holes" in the partition, its evaluation is the greatest integer L such that the set of consecutive integers $[1, L]$ is in the partition, i.e. we stop counting at the first hole.

A level 1 maximization example is presented in Figure 2. The leftmost tree illustrates the start. A Monte-Carlo playout is performed for all 3 possible decisions. At the end of each playout a reward is given, and the decision with the best reward is chosen. This new decision is performed leading to a new state, and will never be backtracked. The process is repeated.

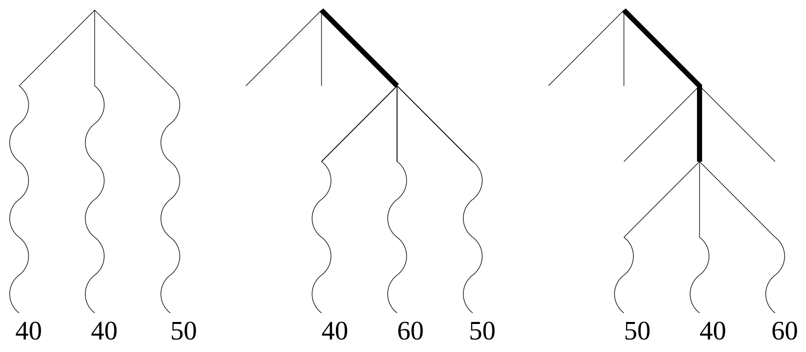

Fig. 2. This figure illustrates three successive decision steps of a level 1 search for a maximization problem, rewards being given at the bottom of branches. Level 1 exploration is represented in thin lines, level 0 Monte-Carlo playouts are shown with wavy lines, decisions are represented with bold lines.

NMC provides a good compromise between exploration and exploitation. It is particularly efficient for one-player games and gives good results even without domain knowledge. However, the results can be improved by the addition of heuristics [15].

Algorithm 1 Nested Monte-Carlo [5]

```
function NMC($position, level$) :
  $best\_score \leftarrow -\infty$
  $best\_playout \leftarrow \{\}$
  while not solution completed do
    if $level = 1$ then
      // collapse level 1 and 0
      // $score\_max, p\_max$: score and playout associated to best branch
      $(score\_max, p\_max) \leftarrow \arg\max_d(\texttt{MonteCarlo}(\texttt{play}(position, d)))$
    else
      $(score\_max, p\_max) \leftarrow \arg\max_d(\texttt{NMC}(\texttt{play}(position, d), level - 1))$
    end if
    if $score\_max > best\_score$ then
      $best\_score \leftarrow score\_max$
      $best\_playout \leftarrow p\_max$
    end if
    $d\_best \leftarrow$ first decision in $best\_playout$
    $position \leftarrow \texttt{play}(position, d\_best)$
  end while
  return $(\texttt{score}(position), best\_playout)$
```

3.2 Nested Rollout Policy Adaptation

The Nested Rollout Policy Adaptation (NRPA) algorithm [17] is inspired by NMC. As in NMC, each level of the algorithm calls a lower level, and each level returns the best score and best playout sequence found at this level. Two main differences are:

– The playout, or rollout, policy is no more a standard Monte-Carlo, but based on a learned policy.
– There is no systematic evaluation of every possible decision associated to any given position, a given number of recursive calls being made instead. So all decisions may not be explored, depending on the policy-based sampling.

The NRPA algorithm is presented in Algorithm 2, where:

– $policy$ is a vector of weights associated to decisions, $policy[x]$ is used to compute the probability of choosing decision x. The initial policy corresponds to a classic Monte-Carlo playout (i.e. equiprobability).
– `code`(p, d_i) returns a domain-specific integer associated to the decision d_i leading from position p to its i^{th} child in the search tree. This integer serves as index in the policy vector. As stated in [17], the function `code`(p, d_i) should preferably be bijective.
– `MonteCarloPolicy`$(position, policy)$ does a playout that differs from classic Monte-Carlo, by choosing the decision not equiprobably but using $policy$. During the playout, a decision is chosen proportionally to $\exp(policy[\texttt{code}(position, i)])$. This function returns a 2-tuple: the evaluation of the terminal position, and the sequence of decisions that were made to obtain it.

In the pseudo-code, the variables $decision$ and $position$, and the function `play`$(position, decision)$ have the same meaning as for the NMC algorithm, see Sect. 3.1.

Algorithm 2 Nested Rollout Policy Adaptation [17]

```
function NRPA ($level$,$policy$)
  if $level$ = 0 then
    // complete position by choosing decisions with probability proportional
    // to $\exp(policy[\texttt{code}(position, i)])$
    return $\texttt{MonteCarloPolicy}(position, policy)$
  else
    $best\_score \leftarrow -\infty$
    for N iterations do
      $(score, playout) \leftarrow$ NRPA $(level - 1, policy)$
      if $score \geq best\_score$ then
        $best\_score \leftarrow score$
        $best\_playout \leftarrow playout$
      end if
      $policy \leftarrow$ Adapt($policy$,$best\_playout$)
    end for
  end if
  return $(best\_score, best\_playout)$

function Adapt ($policy$,$playout$)
  $position \leftarrow$ root
  $policy' \leftarrow policy$
  for each decision $d$ in $playout$ do
    // increase probability of best decision
    $policy'[\texttt{code}(position, d)] \leftarrow policy'[\texttt{code}(position, d)] + \alpha$
    // compute normalization factor
    $z \leftarrow \sum_i exp(policy[\texttt{code}(position, i)])$ over all possible decisions $i$ at $position$
    for each decision $i$ at $position$ do
      $policy'[\texttt{code}(position, i)] \leftarrow policy'[\texttt{code}(position, i)] -$
                                   $\alpha \times \exp(policy[\texttt{code}(position, i)])/z$
    end for
    // prepare to iterate on next decision in best playout
    $position \leftarrow \texttt{play}(position, d)$
  end for
  return $policy'$
```

The NRPA learning is comparable in its motivations with that done in the Upper Confidence Tree search algorithms [7, 14, 16] and consists in increasing probabilities of good decisions while decreasing those for bad decisions. As pointed out in [6], a drawback of the learning is the convergence to local optima.

4 Experimental results

First, we detail the adaptation of the two Monte-Carlo heuristics to the problem at hand, then we report the results obtained on the $WS(5)$ and $WS(6)$ problems.

4.1 Experimental settings

It has been shown in [11, 15] that it is often possible to improve the performance of NMC by adding specific knowledge. A usual way is to bias the probability of decision choice, based of expert knowledge. Having such a probability, rather than a deterministic expert-based choice, may be important to keep diversity [14].

As seen in Section 3.2, the NRPA algorithm learns a policy. We encountered difficulties with the `code`(p, d_i) function. As stated in Sect. 2.3, the number of possible positions (i.e. weakly sum-free partitions under construction) is already too huge for $WS(4)$ to allow the storage of a policy with a bijective code function. It is even more huge when exploring partitions for $WS(5)$ and $WS(6)$. Thus we have chosen to use a 2-dimensional vector of size $WS(k) \times k$, that stores for each integer and each subset the weight associated to the probability of putting this integer in this subset. So the function `code`$(position, decision)$ considers, as $position$ argument, only the current integer to be placed, rather than the whole partition information. So `code` is not bijective. One could perhaps consider using a larger storage with a hashing function to take into account more information from the partition.

We had disappointing results with the standard NRPA algorithm, thus we tried to add expert knowledge. Intuitively, adding expert knowledge in the rollout is harder, as it needs to coexist with the learning and not to interfere with it. To implement this, instead of always making decisions according to the learned policy, we choose with probability $\frac{1}{2}$ a decision according only to the expert knowledge. This allows to take into account both learning and expert knowledge.

For both NMC and NRPA algorithms, a natural implementation is to put integers in the ascending order (i.e. we look for placing integer 1 in a subset, then we want to place integer 2 and so on). Another possibility, which comes from the constraint programming field, is to choose the most constrained integer as the next integer to place. This has the benefit of cutting dead branches of the search tree earlier, thus focusing the search on more promising ones. This significantly improves the search.

Expert knowledge used in our experiments is based on the known optimal partitions of $WS(4)$ and $WS(5)$: as pointed out in [8, 9], all $WS(4)$ solutions are extensions of 2 of the 3 optimal partitions for $WS(3)$. Assuming that this property may hold, more or less strictly, for k-partitions with a larger k, we fix the 23 first integers, with an exception for 16 which can be placed either in subset 1 or 3. From the same knowledge we forbid integers lower than $WS(4) + 1 = 67$ from the last two subsets (subsets 5 and 6). In [12], the best known solution so far showing $WS(6) \geq 581$ has integer 196 as the smallest member of the sixth subset, thus we also ban all integers lower than 196 from subset 6. The lowest allowed integer in the last subset is then greater than or equal to 196. It is very noticeable, as pointed out by [12], that subsets of partitions often contain intervals of consecutive numbers. Thus we add a 90% probability that an integer is put in the first subset (in increasing order) where its immediate predecessor or successor already stays.

Another knowledge from [8] is, for subsets 5 and 6, to try to build sequences of shape $\{a\} \cup [a+2, \ldots, 2a+1]$, with a the first (lowest) integer of the subset. It then seems interesting to place integer $a+1$ in the first subset. We add a 90% probability to play out each of these decisions (i.e. each integer placement).

4.2 Results

We ran 30 independent executions of a level 3 Nested Monte-Carlo search, with embedded expert knowledge as explained above.

Table 1. A weakly sum-free 6-partition of $[1, 582]$.

```
{
  { 1-2, 4, 8, 11, 22, 25, 31, 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, 531, 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, 527-530, 532-535, 570-578},
  { 24, 26-30, 32-36, 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, 371-392}
}
```

A weakly sum-free 6-partition of the integer set $[1, 582]$ was found each time, after an averaged number of $624\,600$ Monte-Carlo sampling, taking on average less than a minute per run on a standard PC. The 6-partitions obtained were usually different on each run. This result yields a new best lower bound for $WS(6)$, namely $WS(6) \geq 582$, and we were not able to obtain a weakly sum-free 6-partition of a larger interval. An example partition showing $WS(6) \geq 582$ is given in Table 1. As for $WS(5)$ we could not increase the current bound of 196, which is its conjectured exact value.

The best result obtained with our knowledge-aware NRPA was 571, while the basic algorithm, without expert knowledge, never attained 300. It remains to experiment if increasing the policy storage could improve on the results.

5 Conclusion

Being now very popular in the AI for games field, Monte-Carlo Tree Search techniques have the ability to address a wider domain. As often, obtaining a successful application relies on the introduction of expert knowledge to foster the potential of the method. It is to be noticed that it is rather easy to embed such knowledge in NMC by simply biasing the sampling probability of solutions. In this study we obtained a new bound for the 6^{th} Weak Schur number, using Cazenave's NMC algorithm and several streamliners to further constrain the problem and reduce its state space. We recall that using these streamliners may constrain too much the search space, thus maybe preventing the discovery of even better bounds.

This work also serves as a validation benchmark for NMC and NRPA. Both heuristics are unable to solve the problem without the addition of expert knowledge. However, it feels more awkward to bias the sampling probabilities in NRPA, since this should normally be done by the policy learning. In our experiments the largest partitions obtained with NRPA have been slightly smaller than those given by NMC. This may be due to the tendency of NRPA to converge to local optima. As possible future works, it may be interesting for the NRPA algorithm to find either a better $code$ function, or another way to incorporate expert knowledge. Hybridization of NMC/NRPA with a constraint solver could also be a promising approach.

References

1. H. Abbott and D. Hanson. A problem of Schur and its generalizations. *Acta Arith.*, 20:175–187, 1972.
2. P. F. Blanchard, F. Harary, and R. Reis. Partitions into sum-free sets. In *Integers 6*, volume A7, 2006.
3. P. Bornsztein. On an extension of a theorem of Schur. *Acta Arith.*, 101:395–399, 2002.
4. T. Brown, B. M. Landman, and A. Robertson. Note: Bounds on some van der Waerden numbers. *J. Comb. Theory Ser. A*, 115(7):1304–1309, Oct. 2008.
5. T. Cazenave. Nested Monte-Carlo search. In C. Boutilier, editor, *IJCAI*, pages 456–461, 2009.
6. T. Cazenave and F. Teytaud. Application of the nested rollout policy adaptation algorithm to the traveling salesman problem with time windows. In Y. Hamadi and M. Schoenauer, editors, *LION*, volume 7219 of *Lecture Notes in Computer Science*, pages 42–54. Springer, 2012.
7. P. Drake. The last-good-reply policy for Monte-Carlo go. *ICGA Journal*, 32(4):221–227, 2009.
8. S. Eliahou, J. M. Marín, M. P. Revuelta, and M. I. Sanz. Weak Schur numbers and the search for G. W. Walker's lost partitions. *Computer and Mathematics with Applications*, 63:175–182, 2012.
9. C. Fonlupt, D. Robilliard, V. Marion-Poty, and A. Boumaza. A multilevel tabu search with backtracking for exploring weak Schur numbers. *Artificial Evolution EA'2011*, volume 7401 of *Lecture Notes in Computer Science*, pages 109–119. Springer, 2012.
10. H. Fredricksen and M. M. Sweet. Symmetric sum-free partitions and lower bounds for Schur numbers. *Electr. J. Comb.*, 7, 2000, Research Paper 32, 9 pages.

11. S. Gelly and D. Silver. Combining online and offline knowledge in uct. In Z. Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 273–280. ACM, 2007.
12. R. Le Bras, C. P. Gomes, and B. Selman. From streamlined combinatorial search to efficient constructive procedures. In *Proceedings of the 15th international conference on Artificial Intelligence*, AAAI'12, 2012.
13. R. Rado. Some solved and unsolved problems in the theory of numbers. *Math. Gaz.*, 25:72–77, 1941.
14. A. Rimmel and F. Teytaud. Multiple overlapping tiles for contextual Monte Carlo tree search. In C. D. Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, C. K. Goh, J. J. M. Guervós, F. Neri, M. Preuss, J. Togelius, and G. N. Yannakakis, editors, *EvoApplications (1)*, volume 6024 of *Lecture Notes in Computer Science*, pages 201–210. Springer, 2010.
15. A. Rimmel, F. Teytaud, and T. Cazenave. Optimization of the nested Monte-Carlo algorithm on the traveling salesman problem with time windows. In C. D. Chio, A. Brabazon, G. A. D. Caro, R. Drechsler, M. Farooq, J. Grahl, G. Greenfield, C. Prins, J. Romero, G. Squillero, E. Tarantino, A. Tettamanzi, N. Urquhart, and A. S. Etaner-Uyar, editors, *EvoApplications (2)*, volume 6625 of *Lecture Notes in Computer Science*, pages 501–510. Springer, 2011.
16. A. Rimmel, F. Teytaud, and O. Teytaud. Biasing Monte-Carlo simulations through rave values. In H. J. van den Herik, H. Iida, and A. Plaat, editors, *Computers and Games*, volume 6515 of *Lecture Notes in Computer Science*, pages 59–68. Springer, 2010.
17. C. D. Rosin. Nested rollout policy adaptation for Monte Carlo tree search. In T. Walsh, editor, *IJCAI*, pages 649–654. IJCAI/AAAI, 2011.
18. I. Schur. Über die kongruenz $x^m + y^m \equiv z^m \pmod{p}$. *Jahresbericht der Deutschen Mathematiker Vereinigung*, 25:114–117, 1916.
19. G. Walker. A problem in partitioning. *Amer. Math. Monthly*, 59:253, 1952.