Meta-Heuristic Search and Square Erickson Matrices

Denis Robilliard, Amine Boumaza, Virginie Marion-Poty

D. Robilliard, A. Boumaza and V. Marion-Poty. Meta-Heuristic Search and Square Erickson Matrices. In the proceedings of the IEEE Congress on Evolutionary Computation (CEC'10), ??–??, 2010

Meta-Heuristic Search and Square Erickson Matrices*

Denis Robilliard and Amine Boumaza and Virginie Marion-Poty

Abstract

A Ramsey theory problem, that can be seen as a 2 dimensional extension of the Van der Waerden theorem, was posed by Martin J. Erickson in his book [1]: "find the minimum n such that if the lattice points of $[n] \times [n]$ are two-colored, there exist four points of one color lying on the vertices of a square with sides parallel to the axes". This was solved recently by Bacher and Eliahou in 2009 [2], who showed that $n = 15$. In this paper we tackle a derived version of this problem, searching for the minimum n that forces the existence of a monochromatic $[3] \times [3]$ subgrid of $[n] \times [n]$ of the form $\{i, i+t, i+2t\} \times \{j, j+t, j+2t\}$ for any 2-coloring of $[n] \times [n]$. We use meta-heuristics on this open problem to find instances of 2-colorations without monochromatic $[3] \times [3]$ subgrid of the above form, setting a lower bound on n. In particular we found such a binary square grid of size 662, implying that $n > 662$.

1 Introduction

Ramsey theory is a branch of mathematics that studies the existence of orderly substructures in large chaotic structures. In fact, the existence of orderly substructures is guaranteed by the theory: "*complete disorder is impossible*[1]". The central question Ramsey theorists struggle to answer is: how large must be the structure such that it contains orderly substructures?

For example, consider a sequence of symbols, colored randomly using a fixed number of colors. Although this sequence may be generated as randomly as possible, if the sequence is sufficiently long there must exist a sub-sequence of a given length that is in arithmetic progression whose symbols are all colored the same.

More formally Van der Waerden's theorem states that considering $k > 1$ symbols (or colors), there exist an integer N such that any k-colored sequence

*Denis Robilliard, Amine Boumaza and Virginie Marion-Poty are with the Univ Lille Nord de France, Laboratoire d'Informatique, Signal et Images de la Côte d'Opale, Maison de la Recherche Blaise Pascal, 50 rue Ferdinand Buisson - BP 719, 62228 CALAIS Cedex, France ; email: {robilliard,boumaza,poty}@lisic.univ-littoral.fr

[1]Quote attributed to Theodore Motzkin.

of length at least N contains a sub-sequence of length $l > 1$ in arithmetic progression (at indices $i, i+t, \cdots, i+(l-1)\,t$) that is monochromatic (i.e. contains only one repeated color). The smallest N for which this holds is called the Van der Waerden number $W(k,l)$. Such numbers are known only for few values of k and l [3]. The 6^{th}, and, largest known, Van der Waerden number for 2 colors was found in 2008 by Kouril et al. [4] and is $W(2,6) = 1132$.

Tibor Gallai [5] extended the above theorem to the d-dimensional case. Within this framework, M. J. Erickson[1] posed the following problem:

Problem 1 *"Find the minimum n such that if the n^2 lattice points of $[n] \times [n]$ are two-colored, there exist four points of one color lying on the vertices of a square with sides parallel to the coordinate axes."*

This problem can be slightly rephrased as the following: "find the smallest n such that there are no monochromatic 2-squares in a binary matrix of size $n \times n$". A monochromatic 2-square is a sub-matrix S with row indices $\{i, i+t\}$ and column indexes $\{j, j+t\}$ with $t \geq 1$ whose 4 corners are colored the same. Following [2] we pose the following definition:

Definition 1 *An Erickson matrix is a binary matrix containing no monochromatic (or constant) 2-squares.*

In the following, we will use the term matrices to refer to binary square matrices.

In [2] the authors give the exact value of $n = 15$ for the above problem. This value was determined by computational means and a clever exploration of the search space, represented as a binary tree and using a special data structure that prevents from storing the tree nodes. Furthermore the authors list all Erickson matrices for $n = 14$.

Similarly to the Van der Waerden numbers, one can search for higher order Erickson matrices i.e. that are free of constant s-squares, where an *s-square* in $[n] \times [n]$ is any square subgrid of the form $\{i, i+l, \ldots, i+(s-1)\,l\} \times \{j, j+l, \ldots, j+(s-1)\,l\}$ with s^2 points.

Definition 2 *For any positive integer $k > 1$, let us denote by $n(k,s)$ the smallest integer n for which any k-coloring of the grid $[n] \times [n]$ contains a monochromatic s-square.*

It follows from a theorem of Gallai [5] that $n(k,s)$ is finite. The actual order of magnitude of $n(k,s)$ is completely unknown at the time of this writing. For instance, before the result $n(2,2) = 15$ of [2], it was only known that: $13 \leq n(2,2) \leq min(W(2,8), 5 \cdot 2^{2^{40}})$ (where $W(2,8)$ is a still unknown Van der Waerden number). Following their discovery of $n(2,2) = 15$, Bacher and Eliahou proposed as open problems the cases $n(3,2)$ and $n(2,3)$.

In the present work, our main goal is to find the largest possible lower bound for $n(2,3)$. More precisely, we aim to provide a lower bound n_0 such that $n(2,3) \geq n_0 + 1$ by constructing a specific matrix of size n_0 which is free of monochromatic 3-squares of the form $\{i, i+l, i+2\,l\} \times \{j, j+l, j+2\,l\}$.

In Section 2, we first focus our interest on the closed $n(2,2)$ problem for which we have the exact bound, in order to compare different search methods on this closely related problem. The algorithm proposed by Bacher and Eliahou in [2]

explores the search space exhaustively starting from a 2×2 matrix constructing successively larger matrices by adding a so-called "elbow" (a column at the right and a row on the bottom). In [4] Kouril and Paul used a dedicated SAT solver to find $W(2, 6)$, a method which could also be used for Erickson matrices, as we show in Section 2.1. However these methods reach their limits when dealing with large matrices. That is why we choose to test heuristic methods. In doing so, we observe that for sizes 13 and 14, the problem of finding Erickson matrices could be of some interest for the evolutionary and meta-heuristics communities as a benchmark problem. Even though the problem is easy for a SAT solver, we found that for these sizes the problem can be difficult for different meta-heuristics within a limited number of evaluations.

Related heuristics have also been applied on similar problems (binary constraint satisfaction problems), such as [6] in a GA framework, or [7] that uses ant colony optimization. In the first case, it was only applied to rather small random instances with a few dozens of variables. In the second case, the pheromone matrix is of size n^2, n being the number of variables. However, in the problem treated in Section 3, we are dealing with matrices of size greater than 600, thus more than 600^2 binary variables. Therefore it seems difficult to use these methods efficiently.

In Section 3, we tackle the problem of finding a largest possible lower bound to $n(2, 3)$. Due to the large computing cost we only apply the best heuristic from the previous experiments. We proceed in an incremental way: once a matrix free of monochromatic 3-squares of a given size is found, we increase the size and run the heuristic again.

2 Searching for Erickson matrices

As our goal is to find lower bounds, we consider a question derived from problem 1, defined as follows:

Problem 2 *"Let $n < 15$ an integer, find a square Erickson matrix of size n."*

This provides us with a range of problems of increasing difficulty, depending on n. From [2] we know the number of solutions[2] to problem 2, denoted Er(n):

[2] This is indeed the number of matrices with upper left coefficient equal to 0: the exact number of solutions is doubled.

n	$\mathrm{Er}(n)$
2	7
3	138
4	5490
5	390856
6	29169574
7	1533415720
8	29085496072
9	156515895928
10	54978562276
11	2510360996
12	1990028
13	570132
14	116114
≥ 15	0

For the smallest values of n, the problem is overly easy due to the small size of the search space, but the density of optimal solutions decreases when n gets closer to 15. For instance, when $n = 14$, which is the largest size for which there exist Erickson matrices, the number of solutions is slightly less than 2^{18} while the search space is of dimension 2^{196}.

2.1 Exact resolution with a SAT solver

Problem 2 can be easily expressed as a boolean satisfiability problem in conjonctive normal form, where the variables are the coefficients of the matrix. For each possible 2-square, we define two clauses that forbid the 2-square to be monochromatic:

$$(x_{i,j} \vee x_{i+t,j} \vee x_{i,j+t} \vee x_{i+t,j+t})$$

$$(\neg x_{i,j} \vee \neg x_{i+t,j} \vee \neg x_{i,j+t} \vee \neg x_{i+t,j+t})$$

where $x_{i,j}$ denotes the matrix coefficient at indices (i, j), and t is the size of the 2-square.

We used the `march_pl` SAT solver developped at Delft University[3]. For the hardest instance, where the matrix size equals 14, the problem was expressed as 1638 clauses on 196 variables, and a solution was found in about 2.5 minutes on a standard PC (this obviously yields solutions for smaller instances). Furthermore we verified the non existence of monochromatic 2-squares on matrices of size 15, where the problem was expressed as 2030 clauses on 225 variables, which could be done in roughly 20 minutes.

However this method cannot be applied on matrices as large as those considered in Section 3. For example, if we express the problem of finding a 300×300 matrix free of monochromatic 3-squares, we obtain 8732592 clauses on 90000 variables, which is untractable with standard SAT solvers. This is why we investigate the performance of heuristic search methods whose behaviour is less dependent on the size of the problem.

[3] `http://www.st.ewi.tudelft.nl/sat`

2.2 Heuristics search methods

We used different search techniques to solve problem 2 that can be classified as either single point search or population based search. On the one hand we applied a local search hybridized with path relinking, a simulated annealing algorithm, and a simulated annealing hybridized with local search. All these heuristics update a search point (a matrix) using a neighborhood operator. Depending on the fitness of the newly generated point and the algorithm used, the new point replaces or not the original one. On the other hand we also applied an evolutionary algorithm, evolving a population of matrices with mutation and crossover. The exact algorithms will be presented shortly, after discussing the features that are common to all of them.

2.2.1 Common features

Initialization method Let us consider the range of possible square binary matrices of any given size, ordered by increasing number of 1 coefficients. Clearly Erickson matrices cannot be located far from the middle of this range: at the extremes, a matrix contains almost only 0's or 1's and thus contains many constant 2-squares. In other words the ratio of 0 coefficients over 1 coefficients in an Erickson matrix cannot be far from 1, i.e. Erickson matrices are almost balanced in 0 and 1. As an example, in size 14 the proportion of balanced matrices is given by the ratio of the binomial coefficient $(14^2, (14^2)/2)$ over the number of matrices 2^{14^2}, which represents 5.7% of the search space but happens to include about 20% of Erickson matrices [8].

Thus, in order to speedup the search, we chose random balanced matrices as initial solutions for all our algorithms.

Fitness function and neighborhood operator The fitness function returns the number of monochromatic 2-squares present in the matrix, thus we work in a minimization context. Our basic neighborhood operator is the simple bit flip. It is also our mutation operator for the genetic algorithm. When a bit flip is performed we only compute the fitness adjustment based on the 2-squares to which the flipped bit belongs. This is more efficient than recomputing the fitness over the whole matrix.

2.2.2 Algorithms details

Local search and the GRASP framework Our local search operator is a hill-climber defined in the following way: if a single bit flip improves the fitness then it is performed, otherwise we search for a second bit-flip such that the combined two flips improve the fitness. Only the bits belonging to monochromatic 2-squares of the current search point (matrix) are considered, since flipping other bits cannot improve the fitness.

On this local search we apply a path relinking heuristic similar to the GRASP (Greedy Randomized Adaptive Search Procedure) method proposed by [9]. We perform the local search on an initial solution, then we explore the path linking this solution to one picked randomly in an elite archive. At each step of the linking path we choose to flip a bit such that the hamming distance to the archive solution is reduced (by one) and the bit-flip yields the better (lowest)

fitness. We tested four archive sizes: 5, 25, 50 and 100. A new elite solution replaces the closest one (with respect to the Hamming distance) with a worse fitness, if any in the archive. Further details about the algorithm can be found in [9].

Simulated Annealing Our Simulated Annealing algorithm (SA) is quite standard (see [10]). We use a static cooling schedule, defined by the following temperature formula:

$$t_k = A/log(1 + (k/B)^C) \tag{1}$$

where k is the iteration counter and A, B, C are constants. This formula provides the initial temperature for $k = 1$. The values for A, B and C are given further in Section 2.3 for the 2-squares problem and in Section 3 for the 3-squares problem. After each decrease of the temperature, we perform $4 \times n^2$ (where n is the size of the matrix) iterations at constant temperature. This number of iterations was chosen by trial and error, in order to allow a potential visit of any point in the search space (potentially flipping any of the n^2 bits of the matrix) as required by the SA equilibrium theory, while not spending too much computations at homogeneous temperature. We remind that the probability of acceptance of a worse search point x' when current point is x is given by:

$$Pr(x \leftarrow x') = exp\left(\frac{fitness(x) - fitness(x')}{t_k}\right)$$

We test three variants of the perturbation operator:

- in the first one, a new search point is generated by randomly flipping one bit of the matrix
- in the second one, we explore only the set of balanced matrices, thus we flip randomly two bits maintaining the matrix balanced
- in the last one, we randomly flip two bits of the matrix.

Large Step Markov Chain (or LSMC) Our third algorithm is inspired from [11]. It can be viewed as a variant of simulated annealing, where a new search point is constructed by:

- first, applying a medium size partly random perturbation (named "kick" in the original article from Martin et al.) to the current point; at this stage the fitness of the new point is probably worse than its predecessor;
- then, optimizing the resulting point with a local search heuristic, here we use the same hill-climbing as in GRASP (see above).

The final point is accepted or rejected as in the simulated annealing method. Thus the main difference with simulated annealing is that we focus on local optima of the local search heuristic. This general framework is also known as *memetic* algorithms [12, 13].

The original algorithm is described in [11], and in our implementation we used the same cooling schedule as in our previous SA method and we tested

two variants. One where the "kick" perturbation was the inversion of 3 random bits belonging to monochromatic 2-squares; and the other where the 3 inverted bits were chosen randomly in the whole matrix. Such "kick" perturbations unfortunately do not guarantee that the loss of fitness remains limited, because of the high level of dependency between coefficients of the matrix. Thus we are not in a context as favourable as in the case of the euclidean TSP studied in [11].

Genetic Algorithm (GA) Our population-based evolutionary algorithm was initially a standard $\lambda+\mu$ Evolution Strategy [14], with deterministic, rank-based, selection. This revealed that such an elitist strategy does not allow the population to escape from local optima. We therefore opted for a Genetic Algorithm framework (see [15, 16]) to explore various selection pressures with tournament selection ranging in size from 4 to 10. The genetic operators are 1-bit mutation as explained above, and three matrix crossovers which we detail now.

The first crossover is the uniform crossover: we create two offspring, each bit of the first offspring has a 1/2 chance of being taken either from the first of the second parent. The remaining bits from the parents are given to the second offspring.

The second crossover is rather standard in its principle, we choose randomly either a column or a row number and we cut two matrices in two parts along that column or row, and then recombine the parts to create two new offspring. Each one combines one part of each parent.

Our third crossover operator creates an offspring (see Figure 1) by taking neighbouring bits alternatively from the two parents. Any given bit originating from one parent has its four neighbours (under the Von Neumann neighbourhood) originating from the other parent. The first offspring takes its upper left corner from the first parent, and the second offspring takes it from the second parent. Notice that two given parents can only produce exactly one pair of offspring. Using this scheme, we intend to preserve all squares of the form $\{i, i+2t\} \times \{j, j+2t\}$ with $t \geq 1$.

2.3 Experiments

In this section we compare the performance of the different search methods. Experiments were conducted with the following setting:

- matrices of size $10 \leq n \leq 14$,
- maximum number of evaluations $5 \cdot 10^7$,
- and 30 independent runs.

The SA and LSMC algorithms used parameters $A = 1.5$, $B = 1 \cdot 10^5$ and $C = 1$. This cooling schedule is illustrated in Figure 2 and was worked out such that the SA stays above $t = 0.2$. We found experimentally that the probability of escaping local extrema was very low when the temperature dropped below 0.2.

The results from the simulated annealing algorithm are in Table 1, those from LSMC are in Table 2 and those from the GRASP method are given in Table 3. We provide only a subset of the GA results for the best population size, 10000,

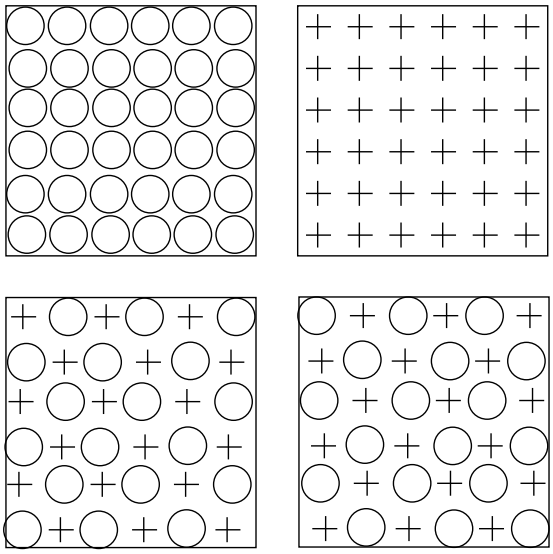

Figure 1: The crossover operator based on the Von Neumann neighbourhood. Two matrices (on the top) generate two offspring (on the bottom).

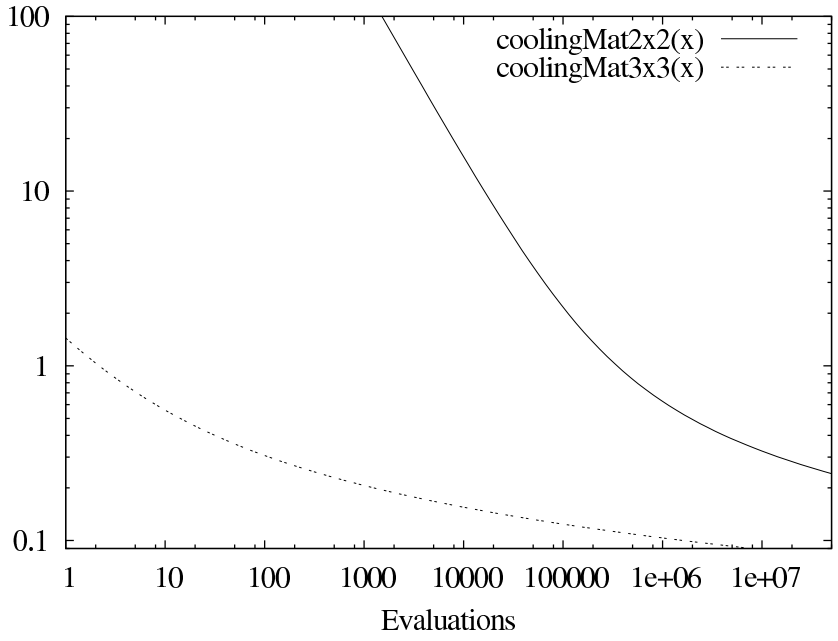

Figure 2: Temperature evolution for SA and LSMC algorithms, according to Equation 1, either for 2-squares labeled "coolingMat2x2" or 3-squares labeled "coolingMat3x3".

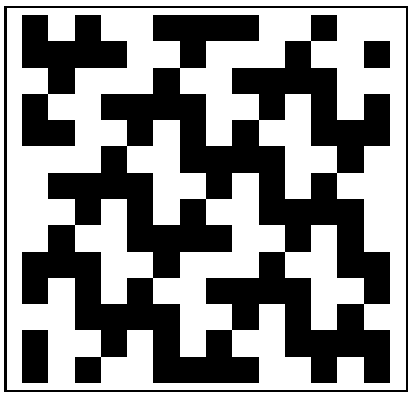

Figure 3: An Erickson matrix of size 14 found by simulated annealing.

and the best set of genetic operators, which turned out to be mutation only (see Table 4). Our experiments showed that all our crossover operators were less efficient than mutation only. We think that these crossovers are probably too disruptive: even though they maintain part of the parent solutions structure, the interdependence of bits is such that this is not enough to prevent the creation of many new constant squares.

From these experiments we can observe that:

• Size 10 is a fairly simple instance, all algorithms found optimal solutions at each run, however with different evaluation cost, LSMC and GA dominating the others, and SA being the most costly.

• In size 11 only the GA fails to find optimal solutions at each run.

• For size 12 an optimal solution is no more found at each run except by the simulated annealing. However some runs of the other algorithms are able to find an optimal solution, GA being the weakest.

• For sizes 13 and 14, only the simulated annealing is able to find optimal solutions (such an optimal size 14 matrix is illustrated in Figure 3).

• The best LSMC variant uses the "kick" perturbation defined randomly on the whole matrix.

• The best simulated annealing variant uses the 1 bit perturbation. It is also to be noted that using a balanced 2 bits perturbation is superior to a random 2 bits change, as it was expected due to the greater density of optimal solutions in balanced matrices. However 1 bit perturbation is still superior.

• The GRASP archive size seems only influential on size 12.

• The GA tournament size does not appear to be strongly decisive, although size 8 seems superior in terms of computational cost.

• Overall, simulated annealing performs the best, although it is more costly in small instances (sizes 10 and 11). GA and GRASP are the weakest heuristics.

Figures 4–7 illustrate the behaviour of the best variant on different problem sizes of each heuristic: GRASP with an archive size of 100, Simulated Annealing with one bit perturbation, LSMC with random kick and GA with population size 10000 and tournament size 8. The curves represent average fitness values over 30 independent runs. We notice that even though GRASP and LSMC start at lower fitness values than SA, their fitness decrease is slower than SA on sizes 12 and up. As it is shown in these figures, GA and GRASP are close and

appears to be the weakest heuristics on these problems.

We do not have strong explanations for the difference in behaviour between these heuristics, although we can suggest that the SA algorithm is more able to escape local optima and thus to perform a better exploration of the search space. It seems plausible that the local search in GRASP and LSMC leads them to over-sample the neighborhood of local optima, explaining the shape of their plots: a promising start followed by a slow improvement. The case of the GA is more intriguing: at first it behaves in a similar fashion to SA, before adopting a GRASP-like behaviour, as illustrated on the plots. It suggests that at that stage the diversity of the population is so reduced as to prevent efficient exploration of the search space.

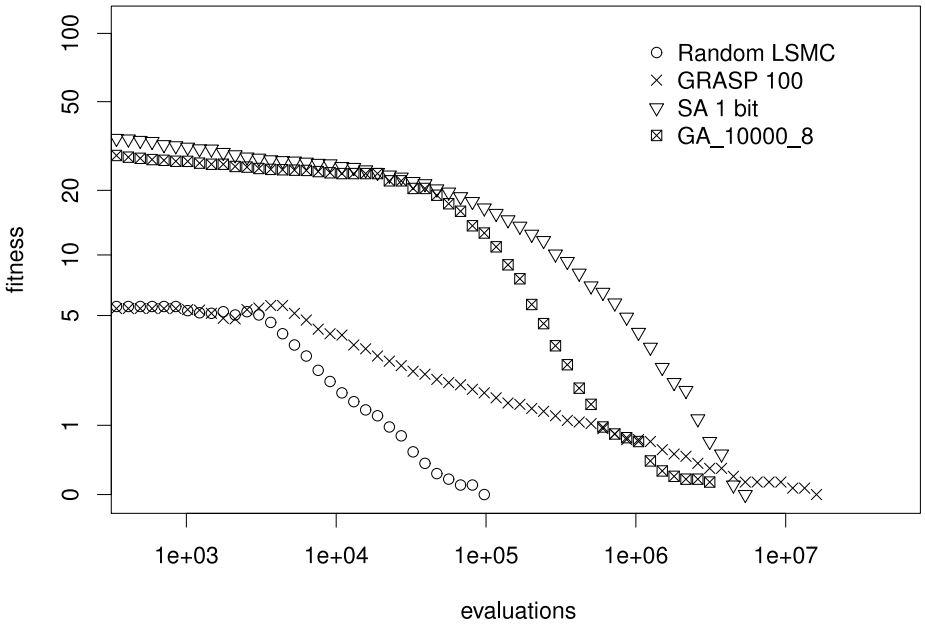

Figure 4: Evolution of the mean fitness for the best variant of the four heuristics for size 11. Both axes are in log scale.

3 A Lower Bound for $n(2,3)$

We can now tackle the search for a lower bound to $n(2,3)$. We remind that we are now searching for square binary matrices, as large as possible, that do not contain any constant 3-squares of the form $\{i, i+l, i+2\,l\} \times \{j, j+l, j+2\,l\}$. The fitness function now returns the number of monochromatic 3-squares present in the matrix, which is to be minimized. First we consider some comparison elements about the fitness landscapes related to $n(2,2)$ and $n(2,3)$, then we report the results of our experiments.

Fitness landscape At first we can observe that a bit in the 4 corners of the matrix obviously belongs to fewer 2-squares (respectively 3-squares) than a bit located closer to the center of the matrix. In order to simplify our reasoning,

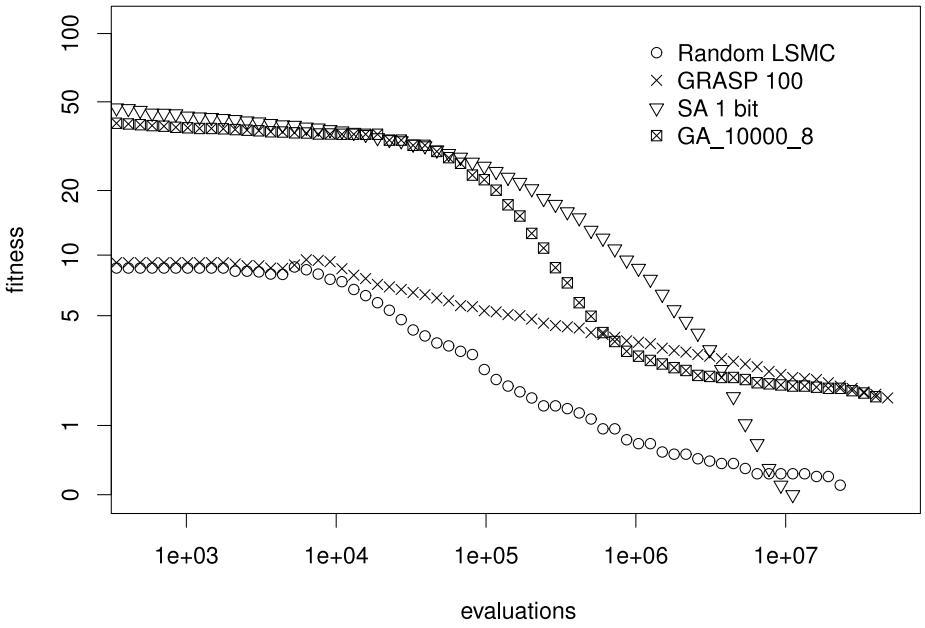

Figure 5: Evolution of the mean fitness for the best variant of the four heuristics for size 12. Both axes are in log scale.

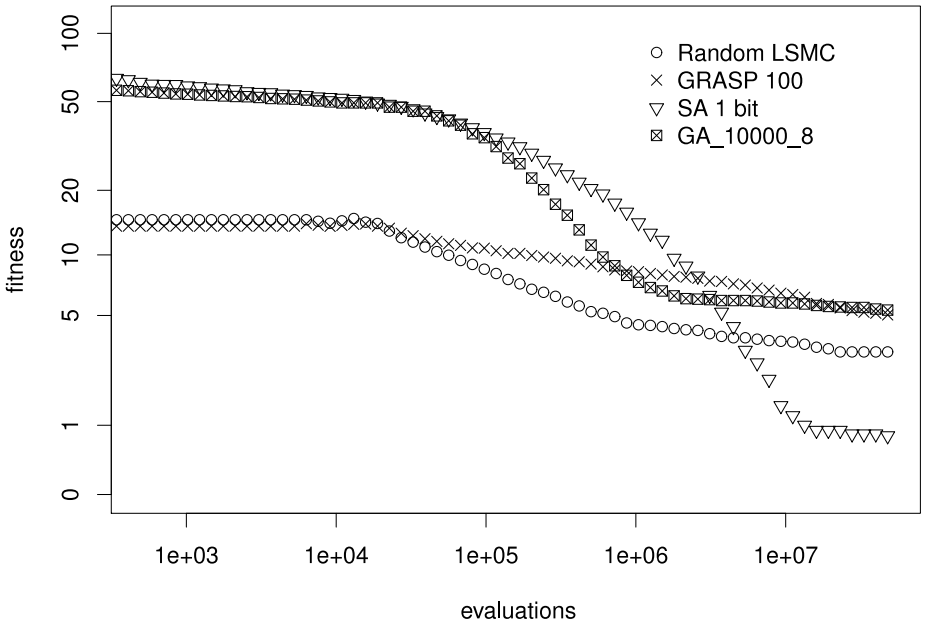

Figure 6: Evolution of the mean fitness for the best variant of the four heuristics for size 13. Both axes are in log scale.

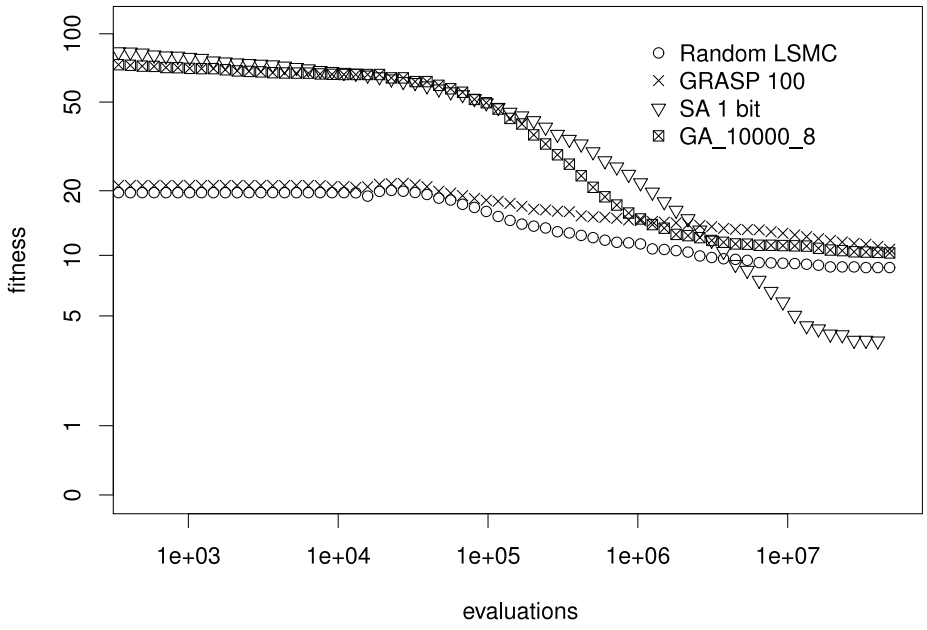

Figure 7: Evolution of the mean fitness for the best variant of the four heuristics for size 14. Both axes are in log scale.

Table 1: Simulated Annealing results for different perturbation operators ("1" for 1 bit mutation, "2*" for balanced 2 bits mutation and "2" for random 2 bits mutation).
Columns are respectively: the matrix size, the perturbation type, the best fitness among 30 runs, the number of runs that reached the best fitness, the mean number of evaluations to reach the best fitness (standard deviation is between parentheses), the mean final fitness for the 30 runs (standard deviation is between parentheses).

size	variant	best	# opt.	opt. cost $\times 10^4$		final fit.	
10	1	0	30	175	(30)	0	(0)
	2*	0	30	187	(52)	0	(0)
	2	0	30	197	(45)	0	(0)
11	1	0	30	363	(70)	0	(0)
	2*	0	30	411	(91)	0	(0)
	2	0	30	420	(93)	0	(0)
12	1	0	30	707	(150)	0	(0)
	2*	0	30	1020	(220)	0	(0)
	2	0	30	1070	(310)	0	(0)
13	1	0	6	2060	(1500)	0.80	(0.41)
	2*	0	6	1350	(810)	0.8	(0.41)
	2	0	3	1220	(740)	0.93	(0.37)
14	1	0	3	2100	(850)	3.67	(2.20)
	2*	0	2	928	(240)	4.83	(1.6)
	2	1	1	1640	(NA)	5.27	(0.94)

Table 2: Large Step Markov Chain results for 2 variants ("2-square" for a kick mutation only on bits belonging to constant 2-squares, "random" for a kick mutation on random bits).
Columns are respectively: the matrix size, the mutation variant, the best fitness among 30 runs, the number of runs that reached the best fitness, the mean number of evaluations to reach the best fitness (standard deviation is between parentheses), the mean final fitness for the 30 runs (standard deviation is between parentheses).

size	variant	best	# opt.	opt. cost $\times 10^4$		final fit.	
10	2-square	0	30	0.31	(0.21)	0	(0)
	random	0	30	0.42	(0.29)	0	(0)
11	2-square	0	30	3.31	(2.4)	0	(0)
	random	0	30	3.26	(2.7)	0	(0)
12	2-square	0	18	856	(1300)	0.433	(0.57)
	random	0	27	419	(680)	0.1	(0.31)
13	2-square	1	1	4040	(NA)	3.5	(1.1)
	random	1	1	1730	(NA)	3.1	(0.84)
14	2-square	7	2	753	(720)	9.27	(1.2)
	random	6	2	1550	(1900)	8.57	(1.6)

we propose to consider the average number of 2-squares and 3-squares to which a bit belongs. This can be computed easily, and we observe that on average the number of 2-squares tends to be twice the number of 3-squares when the matrix size grows, as illustrated in Figure 8.

In terms of fitness landscape, this shows that the inter-dependence between bits (also called *epistasis*) is lower when trying to get rid of monochromatic 3-squares than when working on 2-squares, and thus the search for optimal matrices of a given size is easier when working on $n(2,3)$ than on $n(2,2)$. However the overall problem difficulty increases due to the large size of the matrices involved in searching an as large as possible bound for $n(2,3)$.

Experiments From the experiments on constant 2-squares reported in the previous section, we selected the simulated annealing with single bit perturbation as the most promising heuristic to search for a lower bound for $n(2,3)$.

In order to reach low fitness values for large matrices, we must adapt the cooling schedule so that the algorithm stays longer at lower temperatures than in the 2-square instances. Thus the temperature parameters are now: $A = 1$, $B = 1$ and $C = 0.7$, defining the curve illustrated in Figure 2. The stopping criterion was modified such that the algorithm now halts when there is no fitness improvement for $2 \cdot 10^8$ evaluations.

We explored matrices of size ranging from 30 and up. The largest matrix avoiding monochromatic 3-squares we found is of size 662 and was obtained in 8 hours on an Itanium-2 processor running at 1.5GHz. It is illustrated in Figure 9[4].

[4]Interested readers may download the matrix data file at

Table 3: GRASP results for different archive sizes.
Columns are respectively: the matrix size, the archive size, the best fitness among 30 runs, the number of runs that reached the best fitness, the mean number of evaluations to reach the best fitness (standard deviation is between parentheses), the mean final fitness for the 30 runs (standard deviation is between parentheses).

size	archive	best	# opt.	opt. cost $\times 10^4$		final fit.	
10	5	0	30	3.31	(3.8)	0	(0)
	25	0	30	3.84	(4.7)	0	(0)
	50	0	30	3.84	(4.7)	0	(0)
	100	0	30	3.84	(4.7)	0	(0)
11	5	0	30	384	(390)	0	(0)
	25	0	30	254	(350)	0	(0)
	50	0	30	242	(230)	0	(0)
	100	0	30	329	(390)	0	(0)
12	5	1	14	2740	(1400)	1.57	(0.57)
	25	0	2	2690	(400)	1.57	(0.73)
	50	0	2	3200	(610)	1.7	(0.75)
	100	0	2	3410	(310)	1.63	(0.72)
13	5	3	1	3840	(NA)	5	(0.9)
	25	3	2	775	(250)	5	(0.1)
	50	3	2	2530	(2000)	5.13	(0.97)
	100	2	1	4320	(NA)	5.03	(1)
14	5	8	2	2570	(2900)	10.6	(1.1)
	25	8	2	3670	(840)	10.7	(1.2)
	50	8	4	2240	(1100)	10.4	(1.3)
	100	8	2	3240	(1400)	10.6	(1.1)

Table 4: GA results for a population size of 10000 and different tournament sizes.
Columns are respectively: the matrix size, the tournament size, the best fitness among 30 runs, the number of runs that reached the best fitness, the mean number of evaluations to reach the best fitness (standard deviation is between parentheses), the mean final fitness for the 30 runs (standard deviation is between parentheses).

size	tourn.	best	# opt.	opt. cost $\times 10^4$		final fit.	
10	4	0	30	44.5	(13)	0	(0)
	6	0	30	35.2	(12)	0	(0)
	8	0	30	28.3	(6.9)	0	(0)
	10	0	30	26.6	(5.6)	0	(0)
11	4	0	27	632	(970)	0.1	(0.31)
	6	0	25	258	(410)	0.17	(0.38)
	8	0	26	112	(52)	0.13	(0.35)
	10	0	22	261	(610)	0.27	(0.45)
12	4	0	2	1170	(1300)	1.9	(1)
	6	0	2	1780	(1000)	1.87	(0.9)
	8	0	5	1690	(1300)	1.67	(0.96)
	10	0	1	215	(NA)	2.23	(0.73)
13	4	3	2	1260	(860)	5.77	(1.1)
	6	3	1	313	(NA)	5.37	(0.96)
	8	3	1	179	(NA)	5.33	(1.1)
	10	3	1	131	(NA)	5.37	(1.2)
14	4	8	1	527	(NA)	11.8	(1.7)
	6	7	1	987	(NA)	10.7	(1.6)
	8	7	1	379	(NA)	10.2	(1.4)
	10	8	1	3010	(NA)	10.1	(1.1)

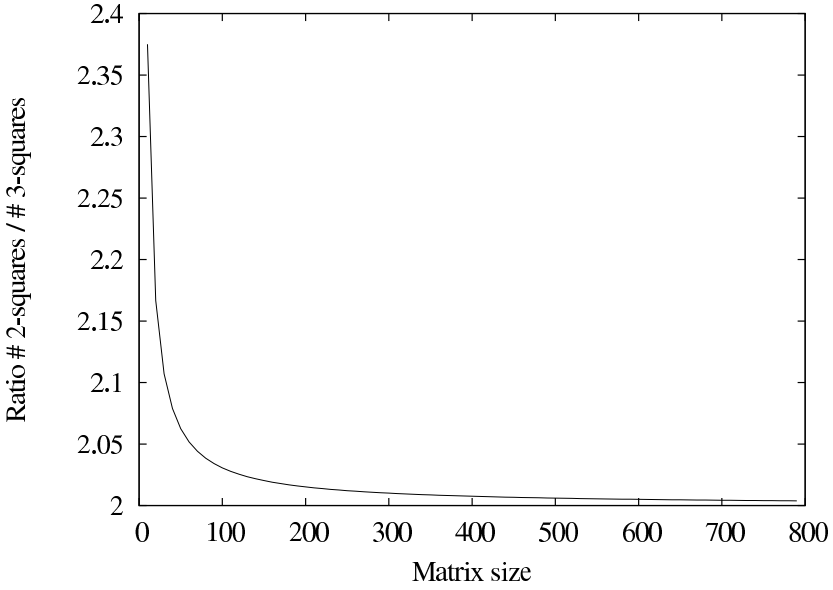

Figure 8: The ratio of possible 2-squares over 3-squares plotted in function of the matrix size.

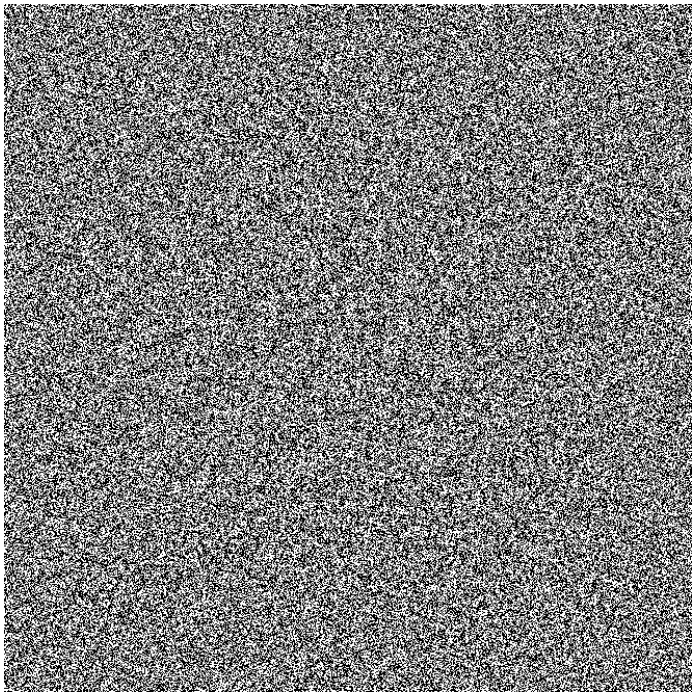

Figure 9: A size 662 matrix free from monochromatic 3-squares.

4 Conclusion

In this paper we aimed at the $n(2,3)$ problem, posed in [2]. Although the recently closed $n(2,2)$ problem can be solved by a standard SAT solver, it is no more the case of $n(2,3)$, due to the large size of the matrices involved. This motivated the use of meta-heuristic methods to find a lower bound to $n(2,3)$, that we first tested on the $n(2,2)$ case, which is related to finding a square Erickson matrix of any given size.

Of the several algorithms we tested, simulated annealing was the most promising, as it was the only one which reached optimal solutions on sizes 13 and 14. From this study we think that searching for Erickson matrices could be an interesting benchmark for the evolutionary and meta-heuristics communities, as it is not so easy to solve. Moreover [2] gives interesting insights on this problem such as the density of optimal solutions for different matrix sizes. Finding a satisfying crossover operator could also constitute an interesting research question.

From these results we chose to apply simulated annealing to the $n(2,3)$ problem, and we found a lower bound: $n(2,3) > 662$. We also gave some insights, based on epistasis, on the relative difficulty of $n(2,2)$ and $n(2,3)$ problems. To our knowledge, this work is the first attempt to provide such a lower bound. In the future we intend to apply self-adaptive cooling schedules to this problem, to avoid having to set the schedule experimentally.

As a closing remark, this study suggests that meta-heuristics which can scale-up to large problem sizes, could be powerful tools to tackle similar problems originating from Ramsey theory.

Acknowledgments

We are grateful to S. Eliahou for introducing us to the fascinating intricacies of Erickson matrices.

References

[1] M. Erickson, *Introduction to Combinatorics*, ser. Series in Discrete Mathematics and Optimization. Wiley-Interscience, 1996.

[2] R. Bacher and S. Eliahou, "Extremal binary matrices without constant 2-squares," Univ Lille Nord de France, Tech. Rep., 2009, to be published in Journal of Combinatorics.

[3] P. Herwig, M. Heule, P. V. Lambalgen, and H. V. Maaren, "A new method to construct lower bounds for van der Waerden numbers," *The Electronic Journal of Combinatorics*, vol. 14, no. 1, p. #R6, 2007.

[4] M. Kouril and J. L. Paul, "The van der Waerden number W(2, 6) is 1132," *Experimental Mathematics*, vol. 17, no. 1, pp. 53–61, 2008.

`www-lisic.univ-littoral.fr/~robillia/matrices/662.txt`

[5] P. Anderson, "A generalization of Baudet's conjecture (van der Waerden's theorem)," *Amer. Math. Monthly*, no. 83, pp. 359–361, 1976.

[6] A. E. Eiben, J. van Hemert, E. Marchiori, and A. G. Steenbeek, "Solving binary constraint satisfaction problems using evolutionary algorithms with an adaptive fitness function," in *Proceedings of the 5th Conference on Parallel Problem Solving from Nature, number 1498 in LNCS*. Springer, 1998, pp. 196–205.

[7] C. Solnon, "Ants can solve constraint satisfaction problems," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 4, pp. 347–357, 2002.

[8] S. Eliahou, 2009, personal communication.

[9] M. Laguna and R. Martí, "Grasp and path relinking for 2-layer straight line crossing minimization," *INFORMS Journal on Computing*, vol. 11, pp. 44–52, 1999.

[10] E. Aarts and J. K. Lenstra, Eds., *Local Search in Combinatorial Optimization*. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[11] O. Martin, S. W. Otto, and E. W. Felten, "Large-step markov chains for the traveling salesman problem," *Complex Systems*, vol. 5, pp. 299–326, 1991.

[12] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms," 1989, caltech Concurrent Computation Program Report 826, California Institute of Technology, Pasadena, USA.

[13] N. J. Radcliffe and P. D. Surry, "Fitness variance of formae and performance prediction," in *FOGA*, 1994, pp. 51–72.

[14] H. Beyer and H. Schwefel, "Evolution strategies - a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, March 2002.

[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. The MIT Press, 1996.