Triangle Packing with Constraint Programming

Patrick Prosser

Computing Science,
Glasgow University, Glasgow, Scotland
pat@dcs.gla.ac.uk

Abstract. In vertex disjoint triangle packing we are given a simple undirected graph G and we have to select the maximum number of triangles such that each triangle is composed of three adjacent vertices and each pair of triangles selected has no vertex in common. The problem is NP-hard and APX-complete. We present three constraint models and apply them to the optimisation and decision problem (attempting to pack $\lfloor n/3 \rfloor$ triangles in a graph with n vertices). In the decision problem we observe a phase transition from satisfiability to unsatisfiability, with a complexity peak at the point where 50% of instances are satisfiable, and this is expected. We characterise this phase transition theoretically with respect to constrainedness. However, when we apply a mixed integer programming model to the decision problem the complexity peak disappears.

Key words: Triangle Packing, constraint programming, phase transition, mixed integer programming, constrainedness

1 Introduction

Given a simple undirected graph $G = (V, E)$ a triangle is a set of three vertices $\{u, v, w\} \subseteq V$ such that $\{\{u, v\}, \{u, w\}, \{v, w\}\} \subseteq E$, i.e. a triangle is three adjacent vertices in the graph. The problem is then to select (*pack*) the maximum number of triangles in the graph such that no two triangles share a common vertex, and is sometimes called a vertex-disjoint triangle packing [5]. There are numerous instances of this problem where we must group compatible objects into sets of three: these objects might be people and the groups teams, or people willing to share a room or work on a project together. A strikingly important variant of this problem is the kidney exchange programs in the United Kingdom [4] and the United States of America [2] where donor/recipient pairs enter a program that allows pair-wise exchanges and cycle-exchanges of length three, but the underlying graph structure is directed with weighted edges.

Triangle packing is NP-hard [12] and APX-complete [17]. There is a variant of the problem where triangles can share vertices but must be edge-disjoint, i.e. edge-disjoint triangle packing. However, in this paper we deal only with vertex-disjoint triangle packing and refer to it simply as triangle packing. Figure 1 shows a simple graph with its triangles tabulated, T_1 to T_5. That is, the graph

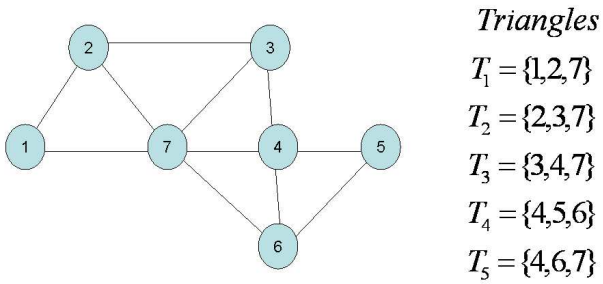


Fig. 1. A simple graph and its triangles.

has $n = 7$ vertices, $m = 5$ triangles, and the largest triangle packing is of size 2, and one such packing is $\{T_1, T_4\}$.

In this paper we explore the triangle packing problem using constraint programming. Three models are proposed, along with variable and value ordering heuristics, and they are compared empirically. We then proceed to the decision problem: for a graph with n vertices can we pack $\lfloor n/3 \rfloor$ triangles? We observe the familiar phase transition behaviour, with a complexity peak at the 50% satisfiability point [7]. The phase transition is then characterised with respect to constrainedness κ [14] and is explored with constraint programming, a pseudo-boolean SAT solver and mixed integer programming. The complexity peak is observed at the crossover point for the constraint program and SAT solver, but when using mixed integer programming the complexity peak disappears entirely, suggesting that the existence of the complexity peak in triangle packing is an algorithmic phenomenon.

2 Constraint Models for Triangle Packing

We present three constraint models for the triangle packing problem. The first model takes a vertex view of the problem and is based on the single-successor model of a graph. The second model takes a triangular view of the problem, and the third and final model is based on the integer linear programming model proposed in [2]. In the subsequent descriptions we assume that we have as input a graph $G = (V, E)$ where $|V| = n$ and vertex $i \in V$. All the constraint models were implemented using the choco toolkit [1]. Further we assume that a variable x has a domain of values $dom(x)$.

2.1 A vertex view

The first model is based on the single successor model of a graph. We have a constrained integer variable for each vertex in the graph. A variable v_i has as a domain of values equal to the set of vertices adjacent to vertex i in G along with the non-existent vertex $n+1$, that is

$$\forall_{i \in V}\ dom(v_i) = \{j | \{i,j\} \in E\} \cup \{n+1\}$$

If a variable $v_i = n+1$ then that vertex is not included in any of the triangles selected. We then have constraints between vertices to force triangles, as follows:

$$\forall_{i,j,k \in V}\ [v_i = j \wedge v_j = k \rightarrow v_k = i]$$

A specialised constraint was implemented in choco to enforce triangles, inspired by the subtour-elimination constraint [6]. This resulted in a more compact model with a linear build time. The smallest domain first heuristic [15] was used for dynamic variable selection, corresponding to a selection of the most isolated vertex. To optimise we maximise the number of variables assigned values less than $n+1$.

2.2 A triangular view

We enumerate the set of m triangles T in G and label the triangles in turn T_1 to T_m. We have a constrained integer variable v_i for each vertex $i \in V$. The variable v_i has as a domain of values the set of triangles in which vertex i participates, with the distinguished triangle T_{m+1} which involves any vertices. That is

$$\forall_{i \in V}\ dom(v_i) = \{x | T_x \in T \wedge i \in T_x\} \cup \{m+1\}$$

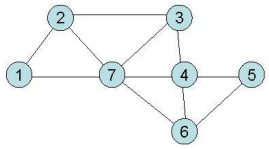
If a variable $v_i = m+1$ this corresponds to vertex i not participating in any selected triangle. A second set of zero/one variables, t_1 to t_m, is introduced to ensure the consistency of triangle selection. The variable $t_j = 1$ if and only if triangle T_j has been selected. This is then maintained with channeling constraints as follows

$$\forall_{i \in V}\ \forall_{j \in dom(v_i)}\ [v_i = j \leftrightarrow t_j = 1]$$

The vertex variables v_1 to v_n are the decision variables. We use the smallest domain first variable ordering, selecting the vertex that participates in the least number of triangles. Value ordering corresponds to the most promising value first [13]. That is for a triangle T_j we count the number of triangles that T_j shares a vertex with, and call this a collision count. Triangles are then sorted in non-decreasing order of collision count. Consequently when a vertex variable is assigned a triangle it will tend to have a small impact on the domain of uninstantiated vertex variables. Finally, to optimise we maximise $\sum_{j=1}^{j=m} t_j$, the number of triangles selected.

2.3 An ILP inspired model

Our final model is inspired by the integer linear programming model proposed in [2]. Again, the m triangles in G are enumerated and an n by m array A is created where $A_{i,j} = 1$ if and only if vertex i participates in triangle j. Figure 2 show the array A for our small sample graph. The j^{th} column of A gives the vertices of G that participate in triangle T_j and the i^{th} row of A gives the triangles in which vertex i participates. We then introduce m zero/one constrained integer



Triangles
$T_1 = \{1,2,7\}$
$T_2 = \{2,3,7\}$
$T_3 = \{3,4,7\}$
$T_4 = \{4,5,6\}$
$T_5 = \{4,6,7\}$

A	T_1	T_2	T_3	T_4	T_5
v_1	1	0	0	0	0
v_2	1	1	0	0	0
v_3	0	1	1	0	0
v_4	0	0	1	1	1
v_5	0	0	0	1	0
v_6	0	0	0	1	1
v_7	1	1	1	0	1

x_1	0/1
x_2	0/1
x_3	0/1
x_4	0/1
x_5	0/1

Fig. 2. The ILP inspired model.

variables x_1 to x_m where $x_j = 1$ if and only if triangle T_j is selected. Constraints are required to ensure that each vertex participates in at most one selected triangle, as follows:

$$\forall_{i \in V} \sum_{j=1}^{j=m} x_j . A_{i,j} \leq 1$$

The decision variables are x_1 to x_m, and since these variables correspond to triangles they are ordered statically in non-decreasing triangle collision order. Value ordering is the value 1 followed by the value 0, preferring to select rather than reject a triangle. Optimisation is via maximising the sum $\sum_{j=1}^{j=m} x_j$.

3 The Optimisation Problem

All experiments were run on a machine with 8 Intel Zeon E5420 processors running at 2.50 GHz, 32Gb of RAM, with version 5.2 of linux. Experiments

were performed over Erdos-Renyi random graphs, $G_{n,p}$, where n is the number of vertices and p is the edge probability. The problem was to pack the maximum number of triangles in the input graph, given a constraint that the maximum is less than or equal to $\lfloor n/3 \rfloor$, i.e. the absolute upper bound on the optimisation is known in advance. Initial experiments showed that the vertex model performed orders of magnitude worse than our other two models and was excluded from our studies. One of the reasons for this is the large number of inherent symmetries within the model. Each triangle generated can be traversed in two directions, and when we have m triangles we have potentially 2^m symmetries. Therefore our first experiment compares the triangular model (Tri) with the integer linear programming inspired model (ILP). Random graphs were generated with $n = 40$, $0.1 \leq p \leq 0.4$, with p varying in 0.01 increments, and 20 graphs generated at each value of p. Figure 3 shows on the left the median number of nodes explored and on the right the median run time in milliseconds.

Our first observation is that the triangular model, Tri, explores fewer nodes but takes more run time and this was the case in all our experiments. However, the most striking features of the graphs is that there is a complexity peak at $p = 0.22$ and this is common to both models. Why should there be a complexity peak? At low values of p few triangles exist and it is easy to find an optimal solution. At high values of p there are many triangles and it is often trivial to find the largest number $\lfloor n/3 \rfloor$ and terminate search. At $p = 0.22$ on average there are 100 triangles to choose from in $G_{40,0.22}$ and the size of the minimum number packed was 11, maximum 13 and average 12 and it appears that it is hard to prove optimality in this region. In Figure 4 we see the complexity peak

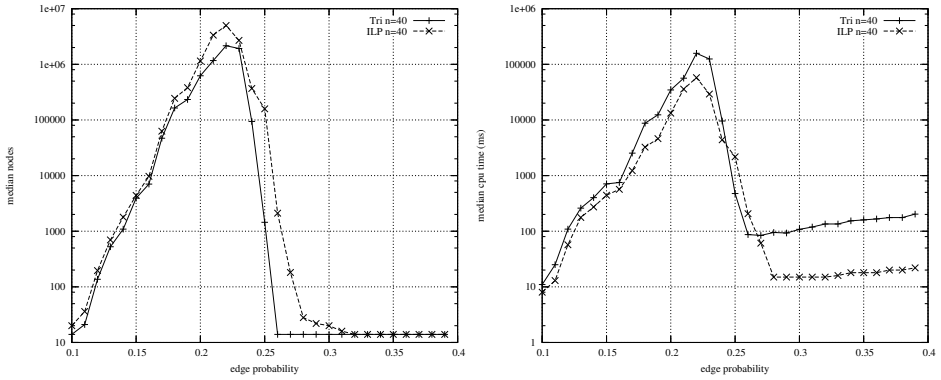


Fig. 3. Search effort for the optimisation problem on $G_{40,p}$ using two constraint models: nodes on the left, cpu time on the right.

in the optimisation problem as we vary n. We show three values of n (21, 30 and 40), with a sample size of respectively 20, 100 and 20 for each value of p. Again

we see a complexity peak, and as n increases the peak becomes more pronounced and occurs at lower values of edge probability p.

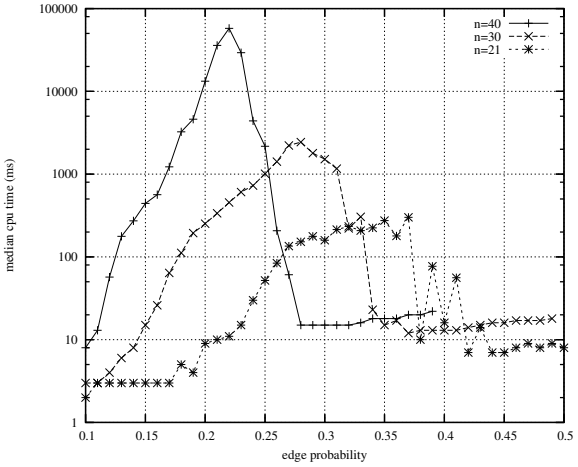


Fig. 4. Optimisation, varying n. As n increases the complexity peak becomes more pronounced and occurs at lower values of edge probability.

4 The Decision Problem

Using the same set of graphs as in the previous study, and the same machine, we now address the decision problem: is there a packing of the input graph that involves $\lfloor n/3 \rfloor$ vertex disjoint triangles? Knowing that the optimisation problem is NP-hard, experience suggests that the decision problem will exhibit a phase transition at some point in its control parameter, in this case edge probability, where instances abruptly change from being unsatisfiable to satisfiable, and when this occurs a complexity peak will emerge [7, 14]. In Figure 5 we see what we expect. On the left are contours for percentage satisfiability against edge probability for four different values of n (n equal to 20, 30, 40, and 50 with a sample size of 100 except for $n = 50$ where a sample size of 50 was used due to excessive run times). On the right we have median run time (in milliseconds) against edge probability for four values of n. We observe a phase transition from unsatisfiable to satisfiable as we increase p and this transition becomes sharper and occurs at lower values of p as we increase n; i.e., we see the familiar easy-hard-easy pattern. [1] It is worth noting that some of these problems were exceptionally hard.

[1] Note that we report run time rather than nodes visited. This is because the choco toolkit records nodes in an integer variable and in some of our $n = 50$ graphs numeric overflow occurred.

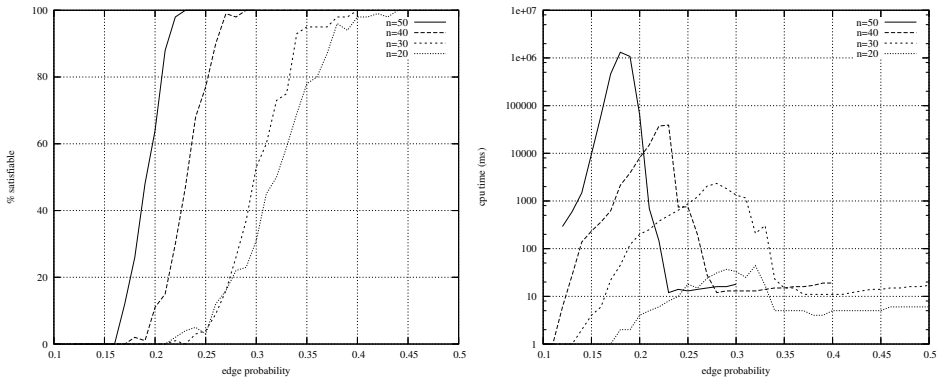


Fig. 5. The decision problem, varying n. On the left percentage solubility and on the right search effort in milliseconds. We see an abrupt phase change and a corresponding complexity peak.

One instance from $G_{40,0.28}$ took in excess of three and a half days CPU time, whereas instances drawn from $G_{50,0.18}$ took on average 64 minutes each.

The phase transition has been characterised in [14] as follows:

$$\kappa = 1 - \frac{log < Sol >}{log|S|}$$

where $< Sol >$ is the expected number of solutions and $|S|$ is the size of the state space. When all states are solutions $\kappa = 0$ and problems are satisfiable and easy, when no state is a solution $\kappa = \infty$ and problems are unsatisfiable and easy, and when there is on average a single solution $\kappa = 1$ and problems are hard. For triangle packing we can compute $|S|$ as follows. First we must compute the expected number of triangles in $G_{n,p}$ and this is

$$\tau = \binom{n}{3} p^3$$

where $\binom{n}{3}$ is the number of ways we can choose three vertices and p^3 is the probability that three vertices are adjacent to each other, and we call this τ. Therefore the size of the state space is the number of ways we can choose $\eta = \lfloor n/3 \rfloor$ triangles from the set of triangles of size τ

$$|S| = \binom{\tau}{\eta}$$

To compute $< Sol >$ we must calculate the probability that a state is a solution, i.e. that having chosen η triangles all triangles contain different vertices. Assume a single triangle has been selected. The probability that the second triangle

selected is compatible with the first is then

$$\frac{\binom{n-3}{3}}{\binom{n}{3}}$$

and the probability that the third triangle selected does not conflict with the first and second is then

$$\frac{\binom{n-6}{3}}{\binom{n}{3}}$$

and so on, in a way similar to that used in the Birthday Paradox where we compute the probability that in a room with n people at least two share the same birthday. Therefore the probability that a state is a solution is then

$$p_{sol} = \prod_{i=1}^{\lfloor n/3 \rfloor - 1} \frac{\binom{n-3.i}{3}}{\binom{n}{3}}$$

and the expected number of solutions is then

$$< Sol >= |S|.p_{sol}$$

In Figure 6 we have on the left percentage satisfiability plotted against κ for n varying from 20 to 50, and on the right median CPU time in milliseconds against κ. From these experiments it appears that the theory of constrainedness

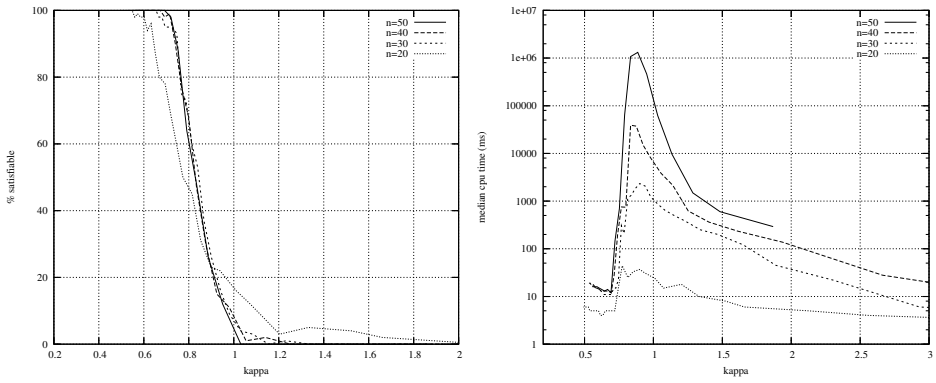


Fig. 6. Percentage solubility (left) and computational effort (right) plotted against constrainedness κ. 50% satisfiability occurs at $0.8 \leq \kappa \leq 0.87$

extends to the triangle packing decision problem, predicting the location of the phase transition with good accuracy.

The triangle packing decision problem was also attacked with minisat+, a pseudo-boolean SAT solver [11]. The resultant model is very similar to the ILP model. A zero/one variable is associated with each triangle, that is $v_i = 1$ if and only if the i^{th} triangle is selected. When triangles i and j share vertices the constraint $v_i + v_j \leq 1$ is posted. Finally there is a constraint to force the selection of $\lfloor n/3 \rfloor$ triangles, i.e. $\sum_{i=0}^{i=m} v_i = \lfloor n/3 \rfloor$. Figure 7 shows a plot of median cpu time against κ. Most obviously, the minisat+ solver is orders of magnitude faster than the constraint programming solver. For $n = 50$ minisat+ is taking seconds to solve whereas the constraint solver takes thousands of seconds. However, both solvers exhibit a computational complexity peak at the 50% satisfiability point. An attempt was made to solve instances at $n = 60$ with a sample size of 50 (values of n less than this have a sample size of 100) but the experiments were aborted after 12 days cpu time, when edge probability reached a value of 0.19, satisfiability approximately 50%, and runtimes typically taking 12 hours per instance.

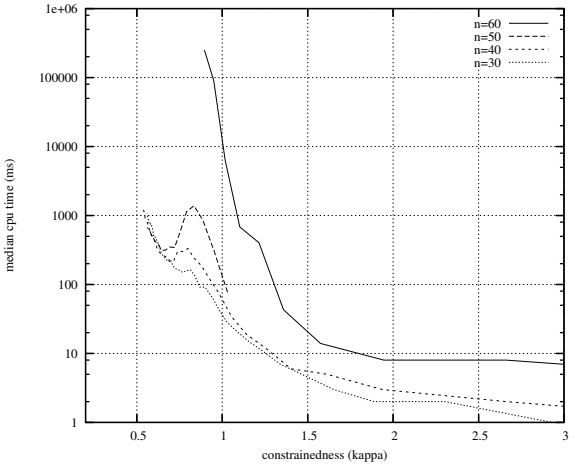


Fig. 7. Median CPU time plotted against constrainedness κ. 50% satisfiability occurs at $0.8 \leq \kappa \leq 0.87$. We see no obvious complexity peak emerging until $n = 50$ and this is located at the 50% point in solubility. Experiments at $n = 60$, sample size 50, were aborted after 12 days.

5 Where have the really hard problems gone?

In 1991 Cheeseman, Kanefsky and Taylor used the provocative title "*Where the really hard problems are*" for their paper on studies of phase transition phenomena in combinatorial problems. They conjectured that all NP-complete decision

problems would exhibit an abrupt phase transition from satisfiable to unsatisfiable as some control parameter was varied and at the phase boundary there would be a complexity peak, and that is where the hard problems are. Subsequent studies confirmed this [16], and also suggested that the complexity peak was a feature of the problem and not of the algorithm, i.e. the complexity peak is algorithm independent. Supporting evidence was given for this in [8] where local search was used. Unsatisfiable instances were filtered out of the problem data set such that a walk-through could be made of the phase transition using local search. Local search encountered a complexity peak at the location of the phase change from satisfiable to unsatisfiable. Experiments were also performed recently using asynchronous digital circuits with feedback loops applied to boolean satisfiability problems [10]. Unsatisfiable instances were again filtered out of the data set and again a complexity peak was encountered at the phase boundary.

Coarfa et al [9] performed experiments on random 3-SAT using a SAT solver (GRASP), mixed integer programming (CPLEX), and a ROBDD-based solver (CUDD). CUDD does not use search, but constructs a symbolic representation of the set of solutions. As clauses were added, increasing the density of instance, GRASP and CPLEX did exhibit a complexity peak but not at the same point. This led to the conclusion that the connection between the crossover point in solubility and the computational complexity peak was not as tight as claimed, and that the peak could well be solver dependent. But could the complexity peak disappear altogether?

The triangle packing decision problem was modeled using the mathematical integer programming (mip) toolkit within Dynadec's COMET, which is SCIP [3], and was applied to the same data sets as before. The model is essentially the ILP constraint model described earlier, and the goal is to maximise the number of triangles packed. If on termination $\lfloor n/3 \rfloor$ triangles have been packed true is delivered, otherwise false. The results of these experiments are shown in Figure 8. The 50% satisfiability point occurs in the range $0.8 \leq \kappa \leq 0.87$. There is no observable complexity peak. There is a steady fall in run time as constrainedness falls, tracking the decrease in the size of the model as edge probability falls. The number of vertices n was increased from 60 to 100 and then onto 200 in the hope that a complexity peak would emerge. No complexity peak emerged.

All results for $n \leq 50$ were compared against the results produced by the constraint programming models and there was complete agreement, ruling out the possibility that there was some error in the experiments at these values of n. At higher values of n we have no way of verifying the results, when instances are unsatisfiable, as we have no data to compare against. However, for $n = 100$, $\kappa = 0.85$ at the 64% solubility point and at $n = 200$, $\kappa = 0.87$ at 42% satisfiability giving us some confidence that the decision problem is being faithfully answered. The mip model was also applied to the hard instance in $G_{40,0.28}$, and terminated after 125 milliseconds, having packed a maximum of 12 triangles, proving that the instance was unsatisfiable. This compares with three and a half days CPU time with our best constraint model. In addition, the experiments for $n = 50$,

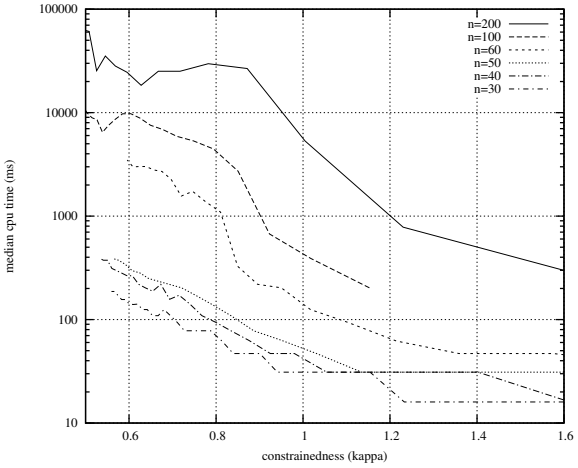


Fig. 8. Median CPU time plotted against constrainedness κ for the mixed integer programming solution. We see no complexity peak.

using a sample size of 50, with $0.1 \leq p \leq 0.3$ in 0.01 increments terminated in less than an hour whilst the constraint programming model took more than a week using 4 processors, i.e. more than a month of CPU time.

One possible criticism is that the problems investigated are too small, and that if larger instances were explored a complexity peak would emerge. Obviously this may be true: just because a complexity peak has not been found is no proof that one does not exist. Nevertheless, it is still interesting to see just how big the problems are that have been explored. One measure used is n, the number of vertices in the graph, and we have explored up to $n = 200$. The true size of an instance is the number of possible triangles in the graph and this is $\tau = \binom{n}{3}p^3$. This is plotted in Figure 9 as the log of the number of possible triangles ($\log(\tau)$) against constrainedness (κ). At low values of κ we have high values of edge probability (p), graphs are dense, and τ is large. Constrainedness increases as we decrease edge probability and τ falls accordingly. At the phase transition point $\kappa \approx 1$ we expect to find hard instance and when $n = 200$ there are on average 660 possible triangles in the graph, and this is really quite large. When $\kappa \approx 0.5$ and $n = 200$ instances have $\tau \approx 6400$, and at $n = 200$ and $\kappa \approx 0.4$ instances have an average $\tau = 24400$. Arguably, these are big instances.

The data was analysed to determine how search cost scales at the 50% crossover point. Median search effort at the 50% point was plotted against n on a log-log scale. The resultant contour has a reasonably straight line with a positive gradient suggesting that search cost scales polynomially with n at the crossover point. Therefore we are left with the awkward question "Where have the really hard problems gone?"

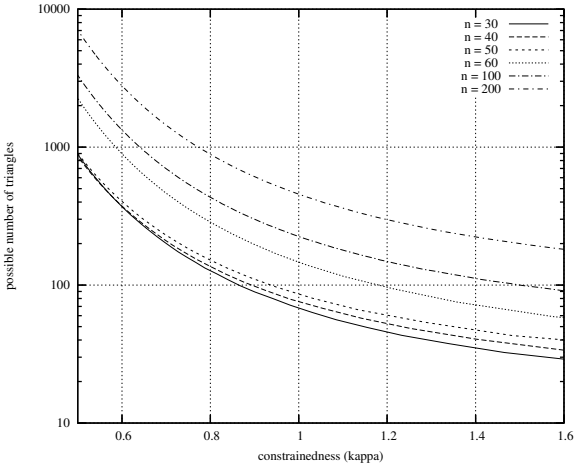


Fig. 9. The number of potential triangles within a graph with n vertices plotted against constrainedness κ. This gives a more realistic measure of the size of a problem when solved using the mixed integer programming model.

6 Conclusion

We have presented a new problem to the constraint programming community, vertex disjoint triangle packing, and have proposed three constraint models with variable and value ordering heuristics. In the optimisation problem we observed a complexity peak, i.e. a region were it was hard to determine optimality. This peak was shared by the different constraint encodings and occurred at all problem sizes investigated.

A decision problem was then presented: pack $\lfloor n/3 \rfloor$ vertices of the graph into vertex disjoint triangles. We saw a sharp phase transition in solubility with a corresponding complexity peak when using a constraint programming solver and a pseudo-boolean SAT solver. This was then characterized with respect to constrainedness κ. The decision problem was then addressed using mixed integer programming and no complexity peak emerged. This is surprising. We expect a complexity peak to occur at or near the phase boundary where 50% of instances are soluble. Although the location of the complexity peak may be an algorithmic phenomenon, when a backtracking search is used a complexity peak occurs nonetheless and tends to be located near the crossover point [9]. But we cannot claim that when using SCIP there is no complexity peak in triangle packing. What we can claim is that in random graphs with 200 or fewer vertices no complexity peak emerges. Going above $n = 200$ is currently beyond the limits of the computational resources available to the author.

As yet we have no explanation why SCIP behaves as it does on triangle packing. One possible line of investigation is to apply SCIP to well studied prob-

lems with classic phase transition behaviour such as number partitioning [2] and boolean satisfiability. If the complexity peak in those problems is encountered when using SCIP then we might conjecture that there is something peculiar about triangle packing. However, if the complexity peak in SAT succumbs to SCIP then this would indeed be interesting and might suggest that the phase transition in NP-complete problems is in fact illusory. In [19] it is argued that the phase transition in combinatorial problems is in fact a continuous process corresponding to the falling number of solutions as an order parameter is increased, and that its abruptness is a consequence of posing the decision problem (in their words "*an existence parameter*") rather than a universal parameter that counts solutions. This is not a new idea. In [18] Smith and Dyer measured the search effort to find all solutions to random constraint satisfaction problems (CSPs). The contour of median search effort to find all solutions blends into the contour for the decision problem at the 50% crossover point, with no complexity peak. To quote from their paper ([18] page 163):

> *If we require to find all solutions to a CSP, or prove that there are none, the median cost decreases smoothly as p_2 (constraint tightness) increases, and nothing noteworthy happens as the problems become insoluble. The phase transition is only an interesting event if just one solution is required: it can then also be viewed as a transition from a partial search of the induced search space (which can be terminated as soon as a solution is found) to a complete search (which is required if there are no solutions).*

The triangle packing problem is similar to independent set in a simple graph. In the first phase of solving we apply a cubic algorithm to enumerate all possible triangles. We then select triangles that do not intersect. This could be modeled using another graph G' where the vertices in G' correspond to the triangles in G and an edge in G' exists if there is an intersection in the vertices of the corresponding triangles in G. Therefore finding an independent set in G' is equivalent to finding vertex disjoint triangles in G, and this is what our ILP-inspired model does, as does our MIP solver. Therefore failure to find a complexity peak in triangle packing when using SCIP is equivalent to failing to find a complexity peak in the corresponding independent set problem. Therefore an interesting next step is to explore the relationship between the phase transition in triangle packing and that in independent set and to investigate the behaviour of the phase transition in independent set as we change solving techniques, as we have done here.

Acknowledgments

I would like to thank: Peter Biro and David Manlove for introducing me to triangle packing and the ILP model; Rob Irving, Ian Gent and Neil Moore for

[2] Preliminary experiments on number partitioning have failed due to technical limitation on numeric accuracy.

help with the mathematics; Ian Gent for pointing me towards [9]; Neil Moore for starting me off on minisat+, Chris Unsworth for the first constraint model and numerous conversations; Pierre Schaus for his help with SCIP and COMET; Ian Gent and David Manlove for reading and commenting on earlier versions of this paper; the CHOCO team for help with the constraint models; Alice Miller for donating countless CPU cycles.

References

1. CHOCO Solver. http://www.emn.fr/x-info/choco-solver/.
2. D.J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter-exchange markets: enabling nationwide kidney exchanges. In *Proc. EC '07: The Eighth ACM Conference on Electronic Commerce, ACM*, pages 295–304, 2007.
3. Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constrained integer programming: a new approach to integrate CP and MIP. In *Proceedings of the 5th international conference on Integration of AI and OR techniques in constraint programming for combinatorial optimization problems (CPAIOR'08)*, pages 6–20, 2008.
4. Peter Biro, David F. Manlove, and Romeo Rizzi. Maximum Weight Cycle Packing in Directed Graphs, with Application to Kidney Exchange Programs. *Discrete Mathematics, Algorithms and Applications*, 1:499–517, 2009.
5. Alberto Caprara and Romeo Rizzi. Packing triangles in bounded degree graphs. *Inform. Proc. Lett.*, 84:175–180, 2002.
6. Yves Caseau and Francois Laburthe. Solving small TSPs with constraints. In *Proceedings International Conference on Logic Programming*, pages 1–15, 1997.
7. Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proceedings IJCAI'91*, pages 331–337, 1991.
8. David A. Clark, Jeremy Frank, Ian P. Gent, Ewan MacIntyre, Neven Tomov, and Toby Walsh. Local search and the number of solutions. In *Proceedings of Principles and Practise of Constraint Programming CP'96*, pages 119–133, 1996.
9. C. Coarfa, D.D. Demopoulos, A.S.M. Aguirre, D. Subramanian, and M.Y. Vardi. Random 3-SAT: The Plot Thickens. *Constraints*, 8:243–261, 2003.
10. W.P. Cockshott, A. Koltes, J.T. O'Donnell, P. Prosser, and W. Vanderbauwhede. A Hardware Relaxation Paradigm for Solving NP-Hard Problems. In *Visions of Computer Science, BCS International Academic Research Conference*, pages 1–12, 2008.
11. Niklas Een and Niklas Sorensson. Translating Pseudo-Boolean Constraints into SAT. *Journal of Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
12. M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Co, 1979.
13. P. A. Geelen. Dual viewpoint heuristics for binary constraint satisfaction problems. In *Proceedings ECAI'92*, 1992.
14. Ian P. Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh. The constrainednss of search. In *Proceedings AAAI'96*, pages 246–252, 1996.
15. R. M. Haralick and G. L. Elliot. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–314, 1980.
16. Brian Hayes. Can't get no satisfaction. *American Scientist*, 85:108–112, 1997.
17. V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inform. Proc. Lett.*, 37:27–35, 1991.

18. Barbara M. Smith and Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.
19. K.A. Zweig, G Palla, and T. Vicsek. What makes a phase transition? Analysis of the random satisfiability problem. *Physica A*, 389:1501–1511, 2010.