Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Getting More Out of the Exposed Structure in Constraint Programming Models of Combinatorial Problems

Gilles Pesant
École Polytechnique de Montréal, Montreal, Canada
CIRRELT, Université de Montréal, Montreal, Canada
gilles.pesant@polymtl.ca

Abstract

To solve combinatorial problems, Constraint Programming builds high-level models that expose much of the structure of the problem. The distinctive driving force of Constraint Programming has been this direct access to problem structure. This has been key to the design of powerful filtering algorithms but we could do much more. Considering the set of solutions to each constraint as a multivariate discrete distribution opens the door to more structure-revealing computations that may significantly change this solving paradigm. As a result we could improve our ability to solve combinatorial problems and our understanding of the structure of practical problems.

Introduction

Many efficient computational methodologies have been designed to solve combinatorial problems. Among those which proceed by building a formal model of the problem, SAT and MIP solvers express theirs in a restricted syntax (Boolean clauses, linear constraints) that allows them to apply powerful algorithms and data structures specialized for that homogeneous form. Constraint Programming (CP) solvers followed a different route, having developed over the years a very rich and heterogeneous modelling language (Beldiceanu et al. 2007) with an extensive set of inference algorithms, each dedicated to a primitive of that language. These high-level primitives, termed global constraints, expose much of the combinatorial structure of a problem. They have been identified over time as being complex enough to provide structural insight yet simple enough for some desired computing tasks to remain tractable. As a result one often finds that an otherwise challenging problem can be modelled using just a few such constraints. The distinctive driving force of CP has been this direct access to problem structure.

$\mathcal{NP}$-hard problems often combine a few "easy" (i.e. polytime-solvable) problems — it is the combination that makes them hard to solve. Consider the well-known TSP: it can be seen as the combination of an assignment problem (minimum weight matching) and of subtour avoidance; or alternately as the combination of the minimum spanning tree problem (strictly speaking, the minimum one-tree) and of limiting the degree of vertices to be at most two. Taken separately, each of these parts is easy. The field of Operations Research has exploited such structural decomposition for a long time through the concept of problem relaxation which provides bounds on the value of the cost function. Lagrangian decomposition provides another opportunity by working on these substructures separately while linking them in the cost function — recent work has adapted this approach to improve communication between constraints in CP (Bergman, Ciré, and van Hoeve 2015; Ha, Quimper, and Rousseau 2015).

A problem to be solved by CP is typically expressed as a Constraint Satisfaction Problem (X, D, C) where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, $D = \{D_1, D_2, \ldots, D_n\}$ a corresponding finite set of domains specifying the possible values that each variable in X can take, and $C = \{c_1, c_2, \ldots, c_m\}$ a finite set of constraints restricting the combinations of values that the variables may take, which can be seen as relations over subsets of X. The exposed structure in CP models, corresponding to each constraint c_j, has been key to the design of powerful filtering algorithms (van Hoeve and Katriel 2006) that identify some variable-value pair (x_i, d) which, given the current domains of the other variables, cannot possibly satisfy c_j and hence filter out d from D_i. To a lesser degree we also exploit that structure to guide search (Boussemart et al. 2004; Pesant, Quimper, and Zanarini 2012) and to explain failure and filtering during search (Stuckey 2010). But we could do much more.

There is a wealth of information inside each of these constraints. In this paper I present a unified perspective that generalizes some of the key features of CP and offers novel powerful ways to exploit combinatorial problem structure.

Solution Sets as Multivariate Distributions

Let's examine the combinatorial structure of individual constraints by considering the set of solutions to a constraint as a multivariate discrete distribution. If we wanted to design algorithms able to query the distribution of solutions to an instance of a constraint, would that be hard? It turns out that for several combinatorial structures we can achieve a lot in polynomial time. What sort of query would we be interested in? Consider the derived marginal distributions over individual variables. Fig. 1 depicts the simple example

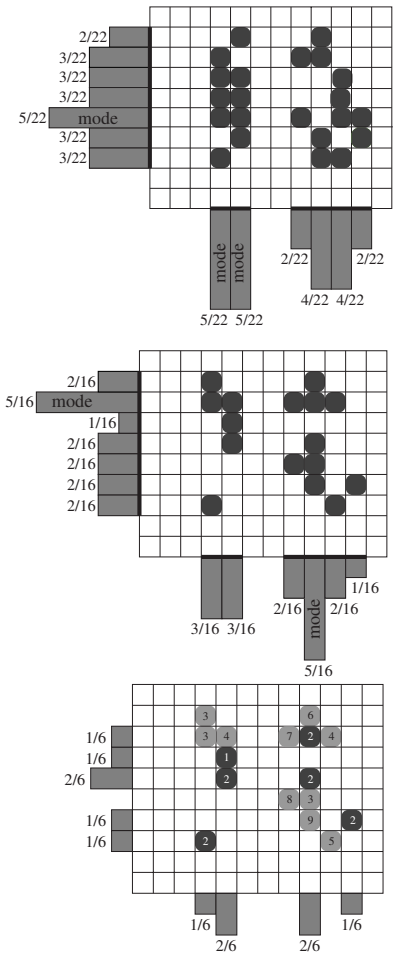

Figure 1: Simple examples of constraints on two variables.

of a constraint on two variables: each row of a grid corresponds to a possible value for the first variable, each column corresponds to a possible value for the second variable, and the dark cells represent the pairs of values that satisfy the constraint (its *solutions*). On the left of and below a grid are depicted the marginal distributions over the first and second variable respectively (i.e. the projection of the set of solutions onto each axis). The core concept of *support* for variable-value pairs in CP precisely corresponds to its namesake for distributions: the subset of its domain whose elements have non-zero frequency (represented as thicker segments on each axis). This concept forms the basis of the definition of *consistency* and its implementation as domain filtering algorithms. If we can compute efficiently and exactly such marginal distributions then we achieve *domain consistency* through their supports. A marginal distribution whose support is a single value identifies a *backbone variable* (Kilby et al. 2005) in a satisfiable CSP i.e. a variable that takes the same value in all solutions. The concept of *solution density* used in counting-based search is equivalent to the marginal distribution, whose *mode* i.e. its most frequent value, corresponds to the branching decision implemented by CP heuristic *maxSD* (Pesant, Quimper, and Zanarini 2012).

Hence some features or statistics of these multivariate distributions already correspond to existing concepts in CP. And more discriminating information is readily available. Returning to the figure, the two constraints depicted in the top and middle grids have very different solution sets, which is reflected in the respective number of solutions (22 vs 16), in the marginal distributions, in their modes, but not in their supports for each variable, which are identical: so if we rely solely on domain consistency the constraints appear equivalent, which they are not. In the context of an optimization problem each solution has a cost, which adds even more structure. The bottom grid shows the same set of solutions as the one in the middle but, because we now add a cost to each solution cell, only the ones whose cost is, say, at most 2 are deemed of interest (darker cells). This yields a very different marginal distribution for each variable, showing the importance of weighing solutions by their relative cost. I outline below further opportunities to exploit the distributions.

Consistency

Domain consistency is the strongest level we can achieve if we operate on the individual domains of variables. Weaker levels of consistency such as bounds consistency identify a superset of the support of marginal distributions over individual variables. Stronger levels of consistency have also been proposed, such as path consistency and k-consistency (Bessiere 2006), but they work *in between* constraints and cannot be reflected solely on individual domains. Now if we consider marginal distributions over subsets of variables (two or more), we may define stronger levels of consistency *within* individual constraints akin to path and k-consistency that can be computed efficiently on our chosen combinatorial structures. Hence such an investigation may give rise to new and possibly more efficient algorithms to achieve already-defined consistency levels or even to new forms of consistency. But information from constraints is traditionally propagated through shared variables' domains, which cannot represent forbidden value combinations for pairs (or larger subsets) of variables. We will come back to this important issue of how to reflect stronger levels of consistencies that cannot be captured in individual domains.

Search

For search we can investigate which features of a distribution should be used to rank the potential branching decisions. Previous work on counting-based search showed that the *maxSD* heuristic, which simply recommends the highest mode from the marginal distributions, works very well in many cases (Pesant, Quimper, and Zanarini 2012). Figures 2 and 3 show the relative performance of that heuristic with respect to other generic heuristics on two problems (see the above reference for a description of these other heuristics). If even this limited exploitation of distributions can improve on state-of-the-art generic search heuristics then a deeper analysis of the distributions may lead to even more

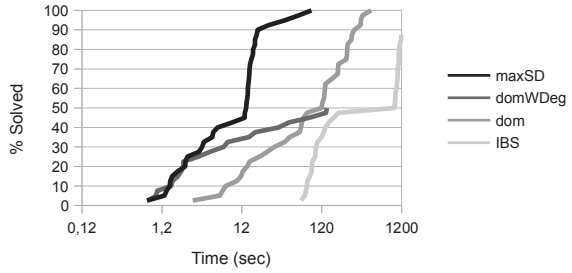

Figure 2: Percentage of Magic Square Completion instances (Problem 19 of the CSPLib) solved after a given time per instance, for a few heuristics (adapted from (Pesant, Quimper, and Zanarini 2012)).

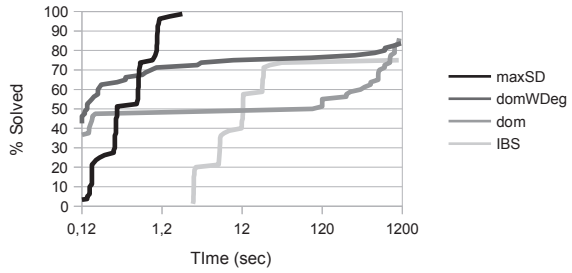

Figure 3: Percentage of Traveling Tournament Problem with Predefined Venues instances (Problem 68 of the CSPLib) solved after a given time per instance, for a few heuristics (adapted from (Pesant, Quimper, and Zanarini 2012)).

robust search.

Solving combinatorial *optimization* problems presents the additional technical challenge of handling distributions over weighted solutions. (Pesant 2016) proposes a way to do this by restricting the distribution to "good" solutions, i.e. that are within some epsilon of a minimum-weight solution. Figure 4 shows the relative performance of a weight-aware variant of the *maxSD* heuristic (denoted *maxSD**) with respect to other generic heuristics on one problem (see the above reference for details). Another option is to apply some damping to solutions according to their distance from the optimum, making better solutions count more. There is considerable potential here when we are confronted to a well-understood combinatorial structure with some side constraints: we can recommend branching decisions that correspond to solution fragments that frequently appear in near-optimal solutions to the "pure" problem (i.e. without the side constraints).

Model Counting

Determining the number of solutions to a CSP (model counting) has a number of applications. Work in enumerative combinatorics is typically limited to "clean" combinatorial structures while work in AI tackles model counting by considering the problem as a whole, e.g. (Ivrii et al. 2016; Gogate and Dechter 2008). Another option here is to count exactly or at least closely for each constraint and then to combine the results, for example by selecting a subset of the

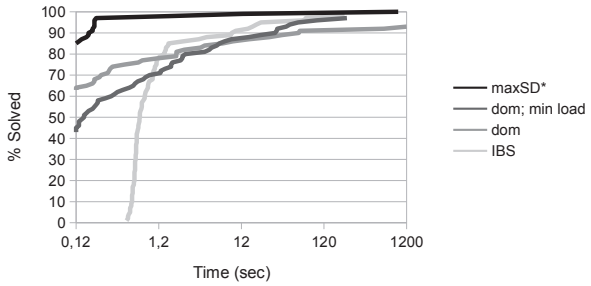

Figure 4: Percentage of Balanced Academic Curriculum Problem instances (Problem 30 of the CSPLib) solved to optimality after a given time per instance, for a few heuristics (adapted from (Pesant 2016)).

constraints forming a partition of the variables and multiplying the results, thus obtaining an upper bound (Pesant 2005).

Uniform Sampling

The ability to sample a complex combinatorial space uniformly at random has important applications such as in test case (or stimuli) generation for software and hardware validation. Knowledge of the marginal distributions over individual variables allows us to draw samples from the set of solutions uniformly at random by considering one variable at a time, deciding which value to assign according to its marginal distribution (viewed as a probability), and then adjusting the marginal distributions to take into account this assignment. If the combinatorial space is described using several structures each with its marginal distributions, the challenge is to adapt this sampling approach in order to preserve as much as possible the uniformity property.

Structural Insights

Starting from the multivariate distribution, which describes the whole structure of the solution set of a constraint, we can take more or less insightful peeks at the structure depending on the nature of our queries. Variable domains are the very flat projections of that distribution on each individual variable. Here are a couple of more revealing ideas.

Learning whether a variable's value is dependent on another variable's value is useful information for search and is related to the concept of *backdoor* (Kilby et al. 2005). While one can sometimes distinguish between main and auxiliary variables by simple inspection of the CP model, in general identifying such relationships proves much harder. We can quantify the amount of dependency between a pair of variables by comparing the marginal distribution over that pair to the product of the marginal distributions over each variable (a perfect match would indicate variable independence).

We already talked about backbone variables, an interesting structural feature but the concept may prove too strict for CP models with non-binary variables. We can relax that concept to *quasi-backbone* variables, when the marginal distribution has a very strong mode or a very narrow head. This may prove more useful in practice.

Querying Distributions

All the above may sound like nice lofty goals that we won't achieve. Typical constraints in CP models involve many more variables and exhibit much more intricate structure than the simple examples at Fig. 1. But because we work with "easy" structures, quite a few tasks can and indeed have been accomplished efficiently. These structures have been selected in CP because the inference task of reaching certain levels of consistency is computationally tractable. The gamble here is that some of the other queries we are interested in are tractable as well. We review next what has been accomplished so far for a few common combinatorial structures.

`alldifferent` constraint. This constraint restricts a set of variables to take pairwise distinct values (Régin 1994). Simply counting the number of solutions to this constraint is intractable ($\#\mathcal{P}$-complete) because it corresponds to computing the permanent of an associated 0-1 matrix, a well-studied problem for which estimators based on sampling and fast upper bounds have been proposed. (Pesant, Quimper, and Zanarini 2012) describe an adaptation of the latter to compute very quickly close approximations of the marginal distributions. This idea can be extended to `global cardinality` constraints (Régin 1996). (Pesant 2016) gives a polytime algorithm based on the above and on solving a minimum-weight bipartite matching problem to compute marginal distributions of good solutions for combinatorial optimization problems.

`regular` constraint. This constraint expresses patterns that must be exhibited in a sequence of variables, as specified by an automaton (Pesant 2004). A slight extension of the data structure used by its domain consistency algorithm suffices to compute exact marginal distributions in polynomial time (Pesant, Quimper, and Zanarini 2012). (Pesant 2016) further extends the latter to compute marginal distributions of good solutions for optimization problems.

`knapsack` constraint. This one expresses linear equalities and inequalities. Exact marginal distributions are computed very similarly to the above but in pseudo-polynomial time; alternatively, approximate marginal distributions based on a relaxation take low polynomial time to compute (Pesant, Quimper, and Zanarini 2012).

`spanning tree` constraint. The identified subset of the edges of a given graph must form a spanning tree of it (Dooms and Katriel 2007). Starting from the Matrix-Tree Theorem, the computation of exact marginal distributions amounts to matrix inversion (Brockbank, Pesant, and Rousseau 2013).

`dispersion` constraint. In order to express balance among a set of variables, the collection of values they take has its mean and its deviation from that mean constrained. It is a generalization of the `spread` and `deviation` constraints. Exact marginal distributions can be computed in polynomial time under reasonable assumptions about the range of the domains (Pesant 2015).

Out of the above a few design patterns have emerged so far. Some constraints already maintain a *compact representation of the solution set* for inference, typically a layered graph in which there is a one-to-one correspondence between paths and solutions. All that is additionally required to compute marginal distributions over single variables is to maintain the number of incoming and outgoing paths at every vertex of that graph. A bigger challenge in this case is how to compute on that graph marginal distributions on pairs of variables or even larger subsets. For constraints on which the counting problem is intractable, *sampling interleaved with constraint propagation* has produced very accurate estimates within reasonable time. Some existing *upper bounds* from the literature, though crude for counting, proved to be quite accurate for marginal distributions (since we take the ratio of two upper bounds) and much faster than sampling. For these two design patterns computing marginal distributions over subsets of variables will be more time-consuming but should not pose additional technical challenges. Lastly for others a *transformation to discrete random variables uniformly distributed over the range of their domain* provides approximate but fast algorithms.

Combining Information

So we have evidence that for several common combinatorial structures the underlying multivariate distribution can be queried efficiently. For some uses, such as consistency, this is sufficient since any value filtering from one structure is valid for the whole CSP. However for others, such as search and sampling, a remaining challenge is how best to combine the information from each structure. As an example the successful *maxSD* branching heuristic for search simply takes the maximum recommendation over all constraints. Should we devise more refined combinations e.g. by considering the way constraints overlap? How can our ability to perform uniform sampling on component structures be used to draw samples from the whole CSP and what would be the statistical properties of that procedure?

Information from constraints is traditionally propagated through shared variables' domains. A promising avenue to investigate is a richer propagation medium. One way to view a variable's domain with respect to a constraint is as a set of variable-value pairs with non-zero frequency, which is rather flat information since all values are on an equal footing. If instead we share marginal distributions over individual variables we can discriminate between values from the perspective of each constraint. And if we go one step further and allow marginal distributions over pairs of variables it makes it easier to propagate stronger levels of consistency. Connections to Belief Propagation (Pearl 1982) should be explored since the nature of the information we propose to share is similar to its messages.

Conclusion

Breaking down a complex combinatorial problem into more manageable and better understood combinatorial structures is foundational in CP. Therefore this solving methodology is very well positioned to exploit these structures. Focusing on multivariate discrete distributions offers many opportunities. I expect that this deeper investigation will have a significant impact on our ability to solve combinatorial problems, getting better solutions faster. It should also provide new insights into the combinatorial structure of practical problems.

Acknowledgements

Financial support for this research was provided by Discovery Grant 218028/2012 from the Natural Sciences and Engineering Research Council of Canada.

References

Beldiceanu, N.; Carlsson, M.; Demassey, S.; and Petit, T. 2007. Global Constraint Catalogue: Past, Present and Future. *Constraints* 12(1):21–62.

Bergman, D.; Ciré, A. A.; and van Hoeve, W. 2015. Improved Constraint Propagation via Lagrangian Decomposition. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, 30–38. Springer.

Bessiere, C. 2006. Constraint Propagation. In *Handbook of Constraint Programming*. Elsevier. 29–83.

Boussemart, F.; Hemery, F.; Lecoutre, C.; and Sais, L. 2004. Boosting Systematic Search by Weighting Constraints. In de Mántaras, R. L., and Saitta, L., eds., *ECAI*, 146–150. IOS Press.

Brockbank, S.; Pesant, G.; and Rousseau, L. 2013. Counting Spanning Trees to Guide Search in Constrained Spanning Tree Problems. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, 175–183. Springer.

Dooms, G., and Katriel, I. 2007. The "Not-Too-Heavy Spanning Tree" Constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, 59–70. Springer.

Gogate, V., and Dechter, R. 2008. Approximate Solution Sampling (and Counting) on AND/OR Spaces. In *Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Proceedings*, volume 5202 of *Lecture Notes in Computer Science*, 534–538. Springer.

Ha, M. H.; Quimper, C.; and Rousseau, L. 2015. General Bounding Mechanism for Constraint Programs. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, 158–172. Springer.

Ivrii, A.; Malik, S.; Meel, K. S.; and Vardi, M. Y. 2016. On Computing Minimal Independent Support and its Applications to Sampling and Counting. *Constraints* 21(1):41–58.

Kilby, P.; Slaney, J. K.; Thiébaux, S.; and Walsh, T. 2005. Backbones and Backdoors in Satisfiability. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, 1368–1373. AAAI Press / The MIT Press.

Pearl, J. 1982. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *Proceedings of the National Conference on Artificial Intelligence*, 133–136. AAAI Press.

Pesant, G.; Quimper, C.-G.; and Zanarini, A. 2012. Counting-Based Search: Branching Heuristics for Constraint Satisfaction Problems. *J. Artif. Intell. Res. (JAIR)* 43:173–210.

Pesant, G. 2004. A Regular Language Membership Constraint for Finite Sequences of Variables. In *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, 482–495. Springer.

Pesant, G. 2005. Counting Solutions of CSPs: A Structural Approach. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 260–265. Professional Book Center.

Pesant, G. 2015. Achieving Domain Consistency and Counting Solutions for Dispersion Constraints. *INFORMS Journal on Computing* 27(4):690–703.

Pesant, G. 2016. Counting-Based Search for Constraint Optimization Problems. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3441–3448. AAAI Press.

Régin, J. 1994. A Filtering Algorithm for Constraints of Difference in CSPs. In *Proceedings of the 12th National Conference on Artificial Intelligence*, 362–367. AAAI Press / The MIT Press.

Régin, J. 1996. Generalized Arc Consistency for Global Cardinality Constraint. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96*, 209–215. AAAI Press / The MIT Press.

Stuckey, P. J. 2010. Lazy Clause Generation: Combining the Power of SAT and CP (and MIP?) Solving. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Proceedings*, volume 6140 of *Lecture Notes in Computer Science*, 5–9. Springer.

van Hoeve, W., and Katriel, I. 2006. Global Constraints. In *Handbook of Constraint Programming*. Elsevier. 169–208.