From Support Propagation to Belief Propagation in Constraint Programming

Gilles Pesant
École polytechnique de Montréal, Montreal, Canada
gilles.pesant@polymtl.ca

July 4, 2018

The distinctive driving force of constraint programming to solve combinatorial problems has been a privileged access to problem structure through the high-level models it uses. From that structure, powerful inference algorithms have shared information between constraints by propagating it through shared variables' domains, traditionally by removing unsupported values. This extended abstract proposes a richer propagation medium made possible by recent work on counting solutions inside constraints.

Many forms of *message passing* algorithms have been investigated in artificial intelligence. One of the earliest and best known, *Belief Propagation* (BP) also known as *sum-product message passing*, was proposed by Pearl (1982) to perform inference on graphical models. From a joint probability distribution it computes approximations of the marginal distributions onto individual variables (i.e. beliefs). Each message is a real-valued function over the domain of a variable that expresses a probability that the variable takes a given value in a model. BP is known to converge to the exact marginal distributions on tree topologies but may not converge in general.

Constraint Propagation can also be viewed as a message passing algorithm, announcing value deletions from domains through the constraint network and necessarily converging because some quantity, namely the size of the Cartesian product of the domains, decreases monotonically (Horsch & Havens, 2013; Werner, 2015). From the perspective of message passing, the deleted values being propagated are messages taking the form of simpler Boolean-valued functions evaluating to false for these deleted values and to true otherwise, which is rather flat information since all non-deleted values are on an equal footing. One way to view a variable's filtered domain with respect to a constraint is as a set of variable-value pairs having non-zero frequency among its solution set — if instead we share the whole frequency distribution over individual variables we can discriminate between values from the perspective of each constraint and, for example, use it for branching in the search tree. Since in general we don't have an explicit description of the solution set of a constraint, that frequency distribution is not readily available and would have to be approximated, perhaps very coarsely. However recent work on solution counting is bringing within reach the computation of exact (or close) distributions for several families of constraints (Pesant, 2017). Once we consider the frequency of a variable-value pair as its likelihood, or probability, of appearing in a solution to the given constraint, we come very close to belief propagation and other message passing algorithms for probabilistic inference.

Among the different types of graphical models, *factor graphs* are closest to constraint networks: they are bipartite graphs featuring variable nodes and factor nodes, the latter corresponding to constraints, with edges between variables and the constraints in which they appear. Messages are sent back and forth between variables nodes and factor nodes, thereby solving in an iterative manner a set of mutually recursive equations defining these messages. Let $\mu_{x\to c}$ denote the message from variable x to constraint c and $\mu_{c\to x}$ the message from constraint c to variable x. We give their definition for BP:

$$\mu_{x\to c}(v) = \prod_{c'\in N(x)\setminus\{c\}} \mu_{c'\to x}(v) \qquad \forall v \in D(x) \tag{1}$$

$$\mu_{c\to x}(v) = \sum_{\boldsymbol{v}:\boldsymbol{v}[x]=v} f_c(\boldsymbol{v}) \prod_{x'\in N(c)\setminus\{x\}} \mu_{x'\to c}(\boldsymbol{v}[x']) \qquad \forall v \in D(x) \tag{2}$$

where $N(x)$ is the neighbourhood of variable x i.e. the constraints in which x appears, $N(c)$ is the neighbourhood of constraint c i.e. its scope, and f_c is the function associated with c, often given as a conditional probability table but in our specific case of a CSP it is better viewed as a decision procedure returning 1 if tuple $\boldsymbol{v}$ satisfies c and 0 otherwise.

The (unnormalized) *marginal* or *bias* θ_x of variable x is computed as

$$\theta_x(v) = \prod_{c \in N(x)} \mu_{c \to x}(v) \qquad \forall v \in D(x)$$

Counting-based search (Pesant, Quimper, & Zanarini, 2012) describes a family of effective branching heuristics in CP. These are expressed through the concept of *solution density*: given a constraint $c(x_1, \ldots, x_k)$, its number of solutions $\#c(x_1, \ldots, x_k)$, respective finite domains D_i $_{1 \leq i \leq k}$, a variable x_i in the scope of c, and a value $v \in D_i$,

$$\sigma(x_i, v, c) = \frac{\#c(x_1, \ldots, x_{i-1}, v, x_{i+1}, \ldots, x_k)}{\#c(x_1, \ldots, x_k)}$$

defines the solution density of pair (x_i, v) in c, measuring how often a certain assignment is part of a solution to c. It is interesting to note that the numerator above corresponds exactly to

$$\sum_{\boldsymbol{v} : \boldsymbol{v}[x] = v} f_c(\boldsymbol{v}), \tag{3}$$

the initial part of the constraint-to-variable message in Equation 2. The solution density can thus be seen as a normalized form of the latter message but *assuming a uniform distribution of values in each domain* since it does not take into account the belief acquired by each variable from the other constraints (i.e. the rest of Equation 2). Summation 3 essentially counts the models (local to c) in which $x = v$. Depending on the constraint and the combinatorial structure it encapsulates, such counting may be intractable (e.g. Valiant, 1979). Nevertheless the recent work on counting-based search has provided efficient algorithms to count either exactly or approximately for several constraints (Pesant et al., 2012; Brockbank, Pesant, & Rousseau, 2013; Pesant, 2015). An efficient implementation of belief propagation with global constraints needs to perform counting weighted by the beliefs about variables, which bears some resemblance to recent work on counting for optimization constraints (Pesant, 2016; Delaite & Pesant, 2017).

This extended abstract promotes a richer propagation medium in CP, made possible by extending such previous work to weighted counting inside constraints and close in spirit to message passing algorithms. This fundamental change can improve our ability to solve CSPs through better-informed branching heuristics but also possibly through the convergence of this new propagation to a particular solution. From the computed marginal distributions, it offers a way to perform uniform sampling of the solution set. It provides structural insights about combinatorial problems by helping to identify dependencies between variables or quasi-backbone variables.

Consider the following example to illustrate our approach:

i. $\texttt{alldifferent}(a, b, c)$

ii. $a + b + c + d = 7$

iii. $c \leq d$

three constraints with variables a, b, c, $d \in \{1, 2, 3, 4\}$. There are two solutions to that CSP: $(a = 2, b = 3, c = 1, d = 1)$ and $(a = 3, b = 2, c = 1, d = 1)$. Even if we enforce domain consistency on each constraint, no filtering occurs. To solve this CSP we would thus be left to branch on variables having identical domains.

Variable a takes value 2 in one of the two solutions, value 3 in one solution as well, and values 1 and 4 in no solution. If we look at the set of solutions as a multivariate discrete distribution, its projection onto a yields $\langle 0, 1, 1, 0 \rangle$ and under the assumption that either solution is equally likely the marginal probability distribution for a is then $\theta_a = \langle 0, 1/2, 1/2, 0 \rangle$. Table 1 gives on the left the marginal for each variable and on

	1	2	3	4
θ_a	0	1/2	1/2	0
θ_b	0	1/2	1/2	0
θ_c	1	0	0	0
θ_d	1	0	0	0

	1	2	3	4
θ_a^i	1/4	1/4	1/4	1/4
θ_a^{ii}	10/20	6/20	3/20	1/20
θ_b^i	1/4	1/4	1/4	1/4
θ_b^{ii}	10/20	6/20	3/20	1/20
θ_c^i	1/4	1/4	1/4	1/4
θ_c^{ii}	10/20	6/20	3/20	1/20
θ_c^{iii}	4/10	3/10	2/10	1/10
θ_d^{ii}	10/20	6/20	3/20	1/20
θ_d^{iii}	1/10	2/10	3/10	4/10

Table 1: True marginals (left) and local marginals (right)

	1	2	3	4
θ_a	.25	.25	.25	.25
θ_b	.25	.25	.25	.25
θ_c	.25	.25	.25	.25
θ_d	.25	.25	.25	.25

	1	2	3	4
θ_a	.50	.30	.15	.05
θ_b	.50	.30	.15	.05
θ_c	.62	.28	.09	.01
θ_d	.29	.34	.26	.11

	1	2	3	4
θ_a	.12	.41	.40	.07
θ_b	.12	.41	.40	.07
θ_c	.84	.15	.01	.00
θ_d	.65	.28	.06	.01

	1	2	3	4
θ_a	.01	.52	.46	.01
θ_b	.01	.52	.46	.01
θ_c	.98	.02	.00	.00
θ_d	.90	.10	.00	.00

Table 2: Initial marginals (top left) and computed marginals after 1st (top right), 5th (bottom left), and 10th (bottom right) iteration

the right the marginals local to each constraint taken individually and over its own set of solutions. We see that from the point of view of the `alldifferent` constraint and for variable a (line θ_a^i) each value is equally likely since it appears in the same number of solutions to that constraint. Whereas for that same variable but from the point of view of the linear equality constraint (line θ_a^{ii}) value 1 is ten times more likely than value 4. Note also that for variable d the two local marginals give conflicting beliefs.

Successful branching heuristics based on local marginals have been proposed before, such as maxSD that branches on the variable-value pair with the highest overall solution density (i.e. local marginal) (Pesant et al., 2012). Given the information in Table 1 it would branch on one of the four variables, assigning it value 1, since overall the highest solution density (marginal) is 1/2, supplied by constraint ii indiscriminately for each variable.

Table 2 presents the evolution of the computed marginals as the number of belief propagation iterations performed increases. Top left are the initial uniform marginals over supported values. Top right is the result of a single iteration: its effect for each variable is to take the average of the local marginals from all constraints in which it appears and it corresponds to previously proposed counting-based branching heuristic avgSD. Then it becomes more interesting as variables send back messages that are not necessarily uniform distributions and constraints compute local marginals weighted by these distributions. The two bottom tables show the marginals computed after five and ten iterations — note how these get closer to the true marginals in Table 1.

To motivate our interest in taking advantage of the high-level modelling typically present in CP, made up of a relatively low number of high-arity constraints, consider Table 3 showing the impact of replacing the `alldifferent` constraint by its decomposition into three disequality constraints: $a \neq b$, $a \neq c$, $b \neq c$. It is much further from the true marginals and does not even appear to converge.

Finally if we add constraint $a \leq b$ there is only one solution left: $(a = 2, b = 3, c = 1, d = 1)$. Again we

	1	2	3	4
θ_a	.29	.41	.25	.05
θ_b	.29	.41	.25	.05
θ_c	.66	.31	.03	.00
θ_d	.48	.38	.12	.02

	1	2	3	4
θ_a	.37	.40	.20	.03
θ_b	.37	.40	.20	.03
θ_c	.61	.37	.02	.00
θ_d	.40	.45	.13	.02

Table 3: Computed marginals after 5th (left) and 10th (right) iteration using a decomposition of `alldifferent`

	1	2	3	4
θ_a	.01	.91	.08	.00
θ_b	.00	.10	.90	.00
θ_c	.99	.01	.00	.00
θ_d	.97	.03	.00	.00

	1	2	3	4
θ_a	.53	.40	.07	.00
θ_b	.29	.30	.37	.04
θ_c	.64	.35	.01	.00
θ_d	.41	.47	.11	.01

Table 4: Computed marginals after 10 iterations with `alldifferent` (left) vs its decomposition (right)

see in Table 4 that the marginals computed from the original model get very close to that solution but not so when using the decomposition.

References

Brockbank, S., Pesant, G., & Rousseau, L. (2013). Counting spanning trees to guide search in constrained spanning tree problems. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 175–183. Springer.

Delaite, A., & Pesant, G. (2017). Counting weighted spanning trees to solve constrained minimum spanning tree problems. In Salvagnin, D., & Lombardi, M. (Eds.), *Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings*, Vol. 10335 of *Lecture Notes in Computer Science*, pp. 176–184. Springer.

Horsch, M. C., & Havens, W. S. (2013). Probabilistic arc consistency: A connection between constraint reasoning and probabilistic reasoning. *CoRR*, *abs/1301.3864*.

Pearl, J. (1982). Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In Waltz, D. L. (Ed.), *Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, August 18-20, 1982.*, pp. 133–136. AAAI Press.

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 Schuurmans, D., & Wellman, M. P. (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pp. 3441–3448. AAAI Press.

Pesant, G. (2017). Getting more out of the exposed structure in constraint programming models of combinatorial problems. In Singh, S. P., & Markovitch, S. (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 4846–4851. AAAI Press.

Pesant, G., Quimper, C., & Zanarini, A. (2012). Counting-based search: Branching heuristics for constraint satisfaction problems. *J. Artif. Intell. Res.*, *43*, 173–210.

Valiant, L. (1979). The Complexity of Computing the Permanent. *Theoretical Computer Science*, *8*(2), 189–201.

Werner, T. (2015). Marginal consistency: Upper-bounding partition functions over commutative semirings. *IEEE Trans. Pattern Anal. Mach. Intell.*, *37*(7), 1455–1468.