Explaining Propagation for Gini and Spread with Variable Mean (Extended Abstract)

Alexander Ek
Dept. of Data Science & AI, Monash University, Melbourne, Australia
CSIRO Data61, Melbourne, Australia
Student (lead) author. All other authors are advisors

Andreas Schutt
CSIRO Data61, Melbourne, Australia

Peter J. Stuckey
Dept. of Data Science & AI, Monash University, Melbourne, Australia

Guido Tack
Dept. of Data Science & AI, Monash University, Melbourne, Australia

Abstract

We introduce two log-linear-time dispersion propagators—(a) spread (variance, and indirectly standard deviation) and (b) the Gini coefficient—capable of explaining their propagations, thus allowing clause learning solvers to use the propagators. Propagators for (a) exist in the literature but do not explain themselves, while propagators for (b) have not been previously studied.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Theory of computation → Constraint and logic programming

Keywords and phrases Spread constraint, Gini index, Filtering algorithm, Constraint programming, Lazy clause generation

Digital Object Identifier 10.4230/LIPIcs.CP.2022.0

Related Version *Full Version*: [2]

Funding This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009.

1 Introduction

In constrained optimisation problems involving multiple agents (stakeholders), we often want to ensure that the solution is balanced and fair. That is, to maximise total utility subject to an upper bound on the statistical dispersion (e.g., spread or the Gini coefficient) of the utility given to different agents or minimise dispersion subject to some lower bounds on utility. These needs arise in, for example, balancing tardiness in scheduling, unwanted shifts in rostering, and desired resources in resource allocation, or minimising deviation from a baseline in schedule repair. These problems are often quite challenging. To solve them efficiently, we want to reason about dispersion effectively. Previous work has studied the case where the mean is fixed, but this may not be possible for many problems, e.g., scheduling where total utility depends on the final schedule. In this paper, we introduce two log-linear-time dispersion propagators—(a) spread [5] (variance, and indirectly standard deviation) and (b) the Gini coefficient—capable of explaining their propagations, thus allowing effective learning solvers to be applied to the above problems. Propagators for (a) exist in the literature but do not explain themselves, while propagators for (b) have not been previously studied. We avoid introducing floating-point variables, which are usually not supported by learning solvers, by

reasoning on integer versions or scaled versions. We show through experimental trials that clause learning can substantially improve the solving of problems where we want to bound dispersion and optimise total utility and vice versa.

The presented propagators filter the lower bound of their dispersion metrics. We only consider filtering the lower bound because usually minimising dispersion or keeping dispersion under some upper bound is of interest. We do not consider propagating the bounds of the X variables being measured, either, since preliminary experimentation indicated this happens very late in the search tree and cannot lead to significant speedups.

An often effective way to solve constrained optimisation problems is the *constraint programming* (CP) solving technology [6]. In CP, each constraint $c \in C$ has a *propagator* f_c, which uses specialised algorithms and logic to reduce the domains of the variables concerning c when invoked, i.e., $f_c(D)(x) \subseteq D(x)$. Only guaranteed non-solutions are removed. CP can be augmented with lazy clause generation (LCG), which combines the techniques of Boolean satisfiability solving (SAT) and CP [4]. A popular LCG solver is Chuffed [1], which we will use and extend in this paper.

Suppose we have an array X of n variables $x_1, \ldots, x_n$ with arithmetic mean μ. The (population) *variance* and the (population) *Gini coefficient* of X are defined, respectively, as follows.

$$\sigma_X^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2, \qquad Gini(X) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}|x_i - x_j|}{n\sum_{j=1}^{n} x_i}.$$

We can introduce constraints $\texttt{spread}(X, M, v, s)$ and $\texttt{Gini}(X, M, g, s)$ for these, respectively. Here, M is constrained to be the sum of X and s is the scaling selected (as to avoid floating-point numbers). For `spread`, v is constrained to be $v = \lfloor \sigma_X^2 \cdot s \rfloor$, i.e., the variance of X, scaled (by s), and rounded down. This means that, e.g., $s = 1$ sets v as a whole number and $s = 100$ sets v as variance in percentage. Note when this is used as a lower bound it is always correct. Similarly, for `Gini`, g is constrained to be $g = \lfloor Gini(X) \cdot s \rfloor$.

2 Formal and Experimental Results

We utilise real-value relaxations for finding lower bounds on spread and Gini. We prove that finding such lower bounds is equivalent to minimising a convex (spread) or quasiconvex (Gini) function, thus can resort to binary chop and arrive at log-linear-time filtering algorithms. We further prove that the generated clauses are correct.

We run two numerical experiments using Chuffed [1] and MiniZinc 2.5.5 [3]. The first one is a simple problem where only minimising dispersion is of concern. The second one is job-shop scheduling problem where multiple agents (owners) submit jobs and compete over the resources (based on [7]). For both dispersion constraints we use four configurations: A decomposition formulation; a simple minimal propagator, which only propagates on the dispersion once all variables X are fixed; and the proposed binary-chop lower-bound propagators with two variations of clause learning, a trivial one and the proposed one. The trivial learning simply uses all bounds of all the variables as explanation, while the proposed one is more frugal.

The results indicate that the proposed spread propagator is the fastest and the best at finding good solutions quickly and proving optimality. The proposed Gini propagator is better at finding good solutions quickly than the other methods, but is slightly outdone by the decomposition in terms of proving optimality.

References

1 Geoffrey Chu. *Improving Combinatorial Optimization.* PhD thesis, Department of Computing and Information Systems, University of Melbourne, Australia, 2011.

2 Alexander Ek, Andreas Schutt, Peter J. Stuckey, and Guido Tack. Explaining propagation for gini and spread with variable mean. In Christine Solnon, editor, *CP'22*, volume 235 of *LIPIcs*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

3 Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a standard CP modelling language. In *CP'07*, LNCS 4741, pages 529–543. Springer, 2007.

4 Olga Ohrimenko, Peter Stuckey, and Michael Codish. Propagation via lazy clause generation. *Constraints*, 14:357–391, 2009.

5 Gilles Pesant and Jean-Charles Régin. SPREAD: A balancing constraint based on statistics. In Peter van Beek, editor, *CP'05*, LNCS 3709, pages 460–474. Springer, 2005. doi:10.1007/11564751_35.

6 Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming.* Elsevier, 2006.

7 Eric Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993.