Decomposition-Based Solving Approaches for Stochastic Constraint Optimisation

David Hemmi
Monash University, Data61/CSIRO, Melbourne, Australia
david.hemmi@monash.edu

Abstract

Combinatorial optimisation problems often contain uncertainty that has to be taken into account to produce realistic solutions. A common way to describe the uncertainty is by means of scenarios, where each scenario describes different potential sets of problem parameters based on random distributions or historical data. While efficient algorithmic techniques exist for specific problem classes such as linear programs, there are very few approaches that can handle general Constraint Programming formulations subject to uncertainty. The goal of my PhD is to develop generic methods for solving stochastic combinatorial optimisation problems formulated in a Constraint Programming framework.

Introduction

Machine learning and statistical inference are popular techniques to forecast customer demand, patient flow in healthcare, or travel times. However, to harness the real value of predictions, one has to understand how to use this information to improve decision making. Importantly, predictions are always subject to a certain confidence level and in order to make realistic decisions, it is thus crucial to take this inherent uncertainty into account.

The focus of my work is on stochastic optimisation problems with a *combinatorial structure* (SCOP), e.g. integer variables and non-linear constraints. *Scenarios* are used to characterise the random variables. Each scenario describes, with a certain probability, the problem when all the random variables are fixed to a specific value. Stochastic problems are composed of multiple stages, the simplest being a two-stage problem. The *first-stage* denotes the problem before information about the random variables is revealed and first-stage decisions are taken with respect to all scenarios. Second-stage decisions are made once the scenario parameters are observed. An example is the stochastic facility location problem, where in the first-stage we have to decide which facilities to open, such that the demand of all customers revealed in the second-stage can be satisfied. I will first introduce a contribution for two-stage problems and then elaborate how the multistage case relates.

Combinatorial optimisation problems can be formulated using Mixed Integer (MIP) or Constraint Programming (CP) technologies. This abstract only addresses CP, as it is most relevant for my research. In CP, modelling frameworks that allow solver agnostic modelling have been developed, e.g. MiniZinc (Nethercote et al. 2007). Problems described in MiniZinc can be solved using standard CP or MIP technology, without tailoring the model for a specific solver. The idea is to separate modelling and solving, such that a model can be expressed without understanding the solving approach used to determine solutions. To solve SCOPs with standard MIP or CP technology one has to formalise the *deterministic equivalent* (DE), a single model composed of all scenarios. The goal of my research is to develop algorithms that can be used to solve scenario-based SCOPs modeled in a problem independent, high-level modelling language, such as MiniZinc.

Background

Multiple methods to solve stochastic programs have been developed in the past (Schultz 2003). The most straight forward method is to formalise the DE and then use standard solving technology to find solutions. The advantage of the DE is its generality and ease of use, yet solving the DE is intractable for all but the smallest problem instances. To solve more challenging SCOPs various research directions have been explored, yet a substantial amount of work has focussed on decomposition algorithms.

Decomposition Methods

An alternative to the DE is to decompose the SCOP, in one of two ways. The problem can either be relaxed by time stages, where a master problem describes the first-stage and contains an approximation of the second stage. This is similar to *Benders decomposition* and named *L-Shaped* method in the context of SCOP. Secondly, the problem can be decomposed by scenarios. For each scenario, a copy of the first-stage variables is introduced. To ensure feasibility, additional consistency constraints that enforce the first-stage variables to agree across all scenarios are added. These consistency constraints can be relaxed by breaking up the model into subparts. Progressive hedging (PH) and the dual decomposition (DD) method are examples that work based on the scenario decomposition, both are complete for linear problems with continuous variables, however not for combinatorial problems (Hemmi, Tack, and Wallace 2017).

Contributions

This section introduces first work by Ahmed (2013) that provides the basis for my contribution that is introduced after.

Evaluate & Cut Ahmed (2013) proposes a scenarios decomposition algorithm for SCOPs with a binary first-stage that works independently of the second-stage structure. Every scenario is modeled as an independent deterministic optimization problem - the result of relaxing the consistency constraints. First, the algorithm solves each scenario independently to optimality, yielding a lower bound and a set of candidates. A candidate is a first-stage variable assignment that can be evaluated against all scenarios, i.e. the first-stage assignment of a specific scenario solution. Secondly, each candidate is evaluated by projecting its variable assignment onto the first-stage variables of all scenarios and solving them again. This yields a feasible solution to the SCOP, as the consistency constraints are satisfied, and an upper bound. Thirdly, by means of *candidate nogoods*, the evaluated candidates are cut off from the search, by adding a nogood constraint to each scenario. Repeating this three steps is guaranteed to find the optimal solution and terminate once the lower bound meets the upper bound, as the first-stage variables have finite support.

Diving Evaluate & Cut (E&C) can be applied to a wide range of problems. An optimality gap is provided during the search and according to Ahmed, early candidate solutions are likely to be the optimal solution to the SCOP. However, the time to prove optimality can be long, as each candidate nogood prunes exactly one complete first-stage assignment. On top of that, the number of evaluation steps required in each iteration is quadratic with the number of scenarios. The scenarios may return distinct candidates that in turn have to be evaluated against all scenarios. We have proposed a framework called *Diving* (Hemmi, Tack, and Wallace 2017), which generates strong nogoods that prune multiple candidates at once. During diving, the partially converged candidates are not evaluated. As a result we avoid the quadratic evaluation step, while remaining complete (able to proof optimality).

After each iteration in E&C the algorithm enters a diving loop with the aim to find *partial nogoods*. In contrast to a candidate nogood, a partial nogood cuts off multiple candidates at once. Consider a SCOP with 5 first-stage variables x_1 to x_5. A candidate nogood contains all first-stage variables, e.g. $c_c = \{x_1 \neq 3 \vee x_2 \neq 6 \vee x_3 \neq 1 \vee x_4 \neq 8 \vee x_5 \neq 3\}$. The added constraint prunes exactly one solution. A nogood composed of only a *subset* of first-stage variables would be much stronger. For example, assume that we can prove that even the *partial* assignment $x_1 = 3 \ \wedge \ x_2 = 6 \ \wedge \ x_3 = 1$ cannot be completed to an optimal solution to the SCOP.

During diving, a set of consistency constraints is enforced, e.g. $x_1 = 3 \ \wedge \ x_2 = 6 \ \wedge \ x_3 = 1$. Then, every scenario is solved independently, yielding a temporary lower bound (with respect to the set of consistency constraints). If the temporary lower bound exceeds the upper bound, it is proven that none of the partially consistent candidates can be extended to a complete first-stage assignment that is better that the incumbent solution. Therefore, a partial nogood excludes all first-stage assignments that are a superset of the partial assignment, is added to the scenarios. If the temporary lower bound remains below the upper bound, an additional consistency constraint is enforced.

Multistage Oftentimes, SCOPs are composed of multiple decision stages. Examples of multistage problems are the management of inventory over multiple weeks with uncertain customer demands, or production planning over a long time horizon with uncertain processing times or due dates. A multistage SCOP can be seen as a sequence of two-stage SCOPS. A first-stage decision is made with respect to all future decisions and scenarios. Thereafter, in the second-stage, a set of decisions that compensates for the first-stage choices is made (as in the two-stage case), yet an additional set of decisions is taken with respect to the proceeding stages. We have shown how to develop a complete recursive algorithm for multistage SCOPs on the basic principles of E&C; obtain and evaluate candidates, then prune solutions (Hemmi, Tack, and Wallace 2018). The recursive algorithm uses bounds in a clever way and remembers candidates with their respective objective value, to speedup the search.

I am the primary author of the mentioned contributions, however I would like to acknowledge and thank my supervisors Guido Tack and Mark Wallace for their guidance and mentorship.

Conclusion

We have introduced diving and shown a substantial performance improvement compared to E&C for the two-stage case. Furthermore, we have shown how to develop a powerful method to solve multistage SCOPs based on the principles of E&C.

To strengthen the diving procedure, various diving heuristics can be implemented. Preliminary results have indicated increased efficiency. Furthermore, it would be interesting to explore whether E&C could be improved using logic based benders cuts.

References

Ahmed, S. 2013. A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters* 41(6):565–569.

Hemmi, D.; Tack, G.; and Wallace, M. 2017. Scenario-based learning for stochastic combinatorial optimisation. In Springer., ed., *CPAIOR*.

Hemmi, D.; Tack, G.; and Wallace, M. 2018. A recursive scenario decomposition algorithm for combinatorial multistage stochastic optimisation problems. In *AAAI*.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. Minizinc: Towards a standard cp modelling language. In *Principles and Practice of Constraint Programming*. Springer. 529–543.

Schultz, R. 2003. Stochastic programming with integer variables. *Mathematical Programming* 97(1):285–309.