Lisbon Wedding: Seating arrangements using MaxSAT

Ruben Martins
Carnegie Mellon University
rubenm@cs.cmu.edu

Justine Sherry
Carnegie Mellon University
sherry@cs.cmu.edu

***Abstract*—Having a perfect seating arrangement for weddings is not an easy task. Can Alice sit next to Bob? Can we ensure that Charles and his ex-girlfriend Eve not be seated together? Meeting such constraints is classically one of the most difficult tasks in planning a wedding – and guests will not accept 'it's NP-complete!' as an excuse for poor seating arrangements. We discuss how MaxSAT can provide the optimal seating arrangement for a perfect wedding, saving brides and grooms (including the authors) from hours of struggle.**

I. Introduction

This benchmark description describes the encoding used for the wedding seating arrangement for our wedding in Lisbon. We needed to seat our guests according to a long list of constraints. For example, members of the same family should sit together; friends who went to school together should sit together; individuals with a history of conflict should be seated apart; etc. We wanted to maximize the happiness of our guests and what better way to do that than to encode the problem into MaxSAT! MaxSAT was an ideal solution for our own wedding: i) it saved us tens of hours, ii) it was stress free, and iii) in the rare case that a guest complained about their seating arrangement, we just blamed the algorithm![1]

II. MaxSAT Encoding

When making a seating arrangement, we first need to define the size of each table and how many guest we have. Assume that our guests are defined by the set P and the tables are defined by the set T. Each table has at least l guests and at most u guests.

Variables. We define our variables as being p_t, meaning that guest p is seated at table t. For simplicity, we do not consider where each person is seated at each table but only if a given person p is seated or not at table t. To characterize our guests, we use a set of auxiliary variables S that denotes characteristics of each person, namely s_t^p denotes the characteristics of person p, seated at table t.

Hard constraints. The hard constraints define the shape of each table and guarantee that each guest will be seated in exactly one place.

- Each guest will be seated at exactly one table:

$$\forall_{p \in P} \sum_{t \in T} p_t = 1$$

- Each table will have at most u guests:

$$\forall_{t \in T} \sum_{p \in P} p_t \leq u$$

- Each table will have at least l guests:

$$\forall_{t \in T} \sum_{p \in P} p_t \geq l$$

Since some guests may have disagreements with each other, we also included some exclusion constraints that guarantee that guests which have conflicts with each other are not seated in the same table. For every pair of guests p and p' that have a conflict with each other we include the following constraints that guarantee that they will not seat together:

$$\forall_{t \in T} (p_t + p'_t \leq 1)$$

To enforce that if a person p is seated at table t then t will contain all labels belonging to p we add additional hard constraints that enforce that table t will contain all the labels from guests that are seated there:

$$\forall_{t \in T} \forall_{p \in P} \forall_{s \in S_p} (p_t \implies s_t^p)$$

Soft constraints. The soft constraints describe the commonalities between guests that share a table. We attach a set of labels to each person that describes her. Example of labels are: spoken languages, university they attended or family last name. Our goal is to minimize the number of labels in each table, i.e. we want to maximize what guests have in common at each table. Let S_t be the set of labels that can occur in table t.

- Minimize the number of labels in each table:

$$min : \sum\nolimits_{t \in T} \sum_{s \in S_t} s$$

Since some labels may be more important than other (e.g. spoken language), we may associate a different weight to each label.

[1]While we were convinced that the algorithm's output was optimal, our guests were not all so enlightened.

III. GENERATOR

We iteratively generated our constraints, adding additional labels or marking guests as in conflict and feeding them to the MaxSAT solver until we arrived at a solution we were happy with. We generated 30 versions of our seating arrangements based on these iterative versions. The generator takes as input: i) the number of tables, ii) the minimum number of guests per table, iii) the maximum number of guests per table, iv) a .csv file with the list of guests and the labels associated with each guest, v) a .txt file with weights for each label, and vi) a .txt file with a set of conflicting labels so that those guests are not seated together.

The problem was encoded using a pseudo-Boolean formalism and translated to MaxSAT using the Open-WBO framework [1]. The following encodings are used by Open-WBO to convert a pseudo-Boolean formula to MaxSAT: i) Ladder encoding [2], [3] (at-most-one constraints), ii) Modulo Totalizer encoding [4] (cardinality constraints) and iii) Generalized Totalizer encoding [5] (pseudo-Boolean constraints).

REFERENCES

[1] Ruben Martins, Vasco Manquinho, Ines Lynce: Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445

[2] Carlos Ansotegui, Felip Manya: Mapping problems with finite-domain variables into problems with boolean variables. SAT 2004: 115

[3] Ian Gent, Peter Nightingale: A new encoding of All Different into SAT. ModRef 2004

[4] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, Hiroshi Fujita: Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers. ICTAI 2013: 9-17

[5] Saurabh Joshi, Ruben Martins, Vasco Manquinho: Generalized Totalizer Encoding for Pseudo-Boolean Constraints. CP 2015: 200-209