Matchmaking with Answer Set Programming

Martin Gebser[1], Thomas Glase[1,2], Orkunt Sabuncu[1], and Torsten Schaub[1]

[1] Universität Potsdam
[2] piranha womex AG, Berlin

Abstract. Matchmaking is a form of scheduling that aims at bringing companies or people together that share common interests, services, or products in order to facilitate future business partnerships. We begin by furnishing a formal characterization of the corresponding multi-criteria optimization problem. We then address this problem by Answer Set Programming in order to solve real-world matchmaking instances, which were previously dealt with by special-purpose algorithms.

1 Introduction

Matchmaking is a form of scheduling that aims at bringing companies or people together that share common interests, services, or products in order to facilitate future business partnerships. The matching process usually starts prior to the actual event and is based on a simple search and offer principle. It involves a registration phase in which matchmaking participants declare what they are looking for and what they have to offer. Based on this information, a human matchmaker, equipped with experience in the community and the business, identifies promising matches and makes meeting proposals to the participants, who can then either accept or decline proposed matches.

In this report from the field, we show how we solved the matchmaking problem for several fairs by means of Answer Set Programming (ASP; [1]). The resulting system is used by the company *piranha womex AG*[3] for computing matchmaking schedules for the world music exposition (WOMEX) and other fairs.

We begin by developing a formal characterization of the matchmaking problem in Section 2. We then address this problem by ASP in Section 3. Section 4 provides an empirical analysis showing that our ASP-based approach allows for solving real-world matchmaking instances that were previously dealt with by special-purpose algorithms.

2 Matchmaking Scheduling

Matchmaking events bring companies or people sharing common interests, services, or products together in order to facilitate future business partnerships. Such events usually take place at occasions like technology, business, or music and entertainment fairs. A match is a face-to-face meeting of two parties. While a party may be a representative of a company, it may also be an individual like an artist or a producer. Given that an individual can be regarded as a company with one representative, it is viable to view matches as meetings between company representatives.

[3] `http://www.piranha.de`

A matchmaking event starts with an initial phase in which interests and offers of participants are analyzed for potential business partnerships. This phase results in a set of matches recommended to participants, and all accepted recommendations constitute the final set of matches to be scheduled. Scheduling each match for a time slot and a location (e.g. a table) yields the matchmaking schedule of an event. In the following, we formalize these intuitions by defining the matchmaking scheduling problem.

Definition 1. *Let T be a linearly ordered set representing time slots. Locations and companies are represented by sets L and C, respectively. A match is a set consisting of two companies from C. Let M be the set of all matches to be scheduled and P be the set of all persons. Each person $p \in P$ works for a company $w(p) \in C$ and has a time preference $\emptyset \subset t(p) \subseteq T$.*

Matchmaking scheduling is the problem of finding a feasible and optimal schedule $S = \langle S_{\mathcal{C}}, S_{\mathcal{P}} \rangle$, where $S_{\mathcal{C}}$ (resp., $S_{\mathcal{P}}$) is a relation from M (resp., P) to $T \times L$. A schedule S is feasible if $S_{\mathcal{C}}$ is a total function, i.e. each match is scheduled once, each location hosts at most one match at a time, i.e.

$$\forall t \forall l \forall m_1 \forall m_2 : S_{\mathcal{C}}(m_1, (t,l)) \wedge S_{\mathcal{C}}(m_2, (t,l)) \Rightarrow m_1 = m_2 \; , \tag{1}$$

and for each company in a match, exactly one employee of the company is scheduled:

$$\forall t \forall l \forall m \forall c \exists! p : S_{\mathcal{C}}(m, (t,l)) \wedge c \in m \Rightarrow S_{\mathcal{P}}(p, (t,l)) \wedge w(p) = c \; , \tag{2}$$

$$\forall t \forall l \forall p \exists m : S_{\mathcal{P}}(p, (t,l)) \Rightarrow S_{\mathcal{C}}(m, (t,l)) \wedge w(p) \in m \; . \tag{3}$$

A feasible schedule S is optimal if it is not dominated w.r.t. the following ranked objectives, ordered by their precedence:

- *The number of overlaps, where a person has more than one match at a time slot, should be minimized:*

$$\min \sum_{p \in P, t \in T, \sigma(p,t) > 0} \sigma(p,t) - 1 \; , \text{ where } \sigma(p,t) = |\{l \in L \mid (p,(t,l)) \in S_{\mathcal{P}}\}| \; . \tag{O}$$

- *The number of matches at unpreferred time slots of persons should be minimized:*

$$\min \sum_{p \in P, t \in T \setminus t(p)} |\{l \in L \mid (p,(t,l)) \in S_{\mathcal{P}}\}| \; . \tag{P}$$

- *The number of idle time slots between matches of companies should be minimized:*

$$\min \sum_{c \in C} \left| \left\{ t \in T \middle| \begin{array}{l} (m_1,(t_1,l_1)) \in S_{\mathcal{C}}, (m_2,(t_2,l_2)) \in S_{\mathcal{C}}, t_1 < t < t_2, \\ c \in m_1, c \in m_2, \{c \in m \mid (m,(t,l)) \in S_{\mathcal{C}}\} = \emptyset \end{array} \right\} \right| \; . \tag{G}$$

- *The number of location changes between consecutive matches of companies should be minimized:*

$$\min \sum_{c \in C} \left| \left\{ t \in T \middle| \begin{array}{l} (m_1,(t,l_1)) \in S_{\mathcal{C}}, (m_2,(s(t),l_2)) \in S_{\mathcal{C}}, l_1 \neq l_2, \\ c \in m_1, c \in m_2, s(t) \text{ is the successor time slot of } t \end{array} \right\} \right| \; . \tag{T}$$

– *Used resources (time slots and locations) should be minimized:*

$$\min\left(|\{t \in T \mid (m,(t,l)) \in S_{\mathcal{C}}\}| + |\{l \in L \mid (m,(t,l)) \in S_{\mathcal{C}}\}|\right). \qquad \text{(U)}$$

Although a person cannot attend several matches at the same time, we do not impose objective (O) as a hard constraint. The rationale of scheduling all matches and tolerating conflicts is to maximize the chances of future business partnerships. In reality, conflicts may be resolved a posteriori, for instance, by sending more personnel to an event.

Note that the above definition of particular objectives and their precedence reflects practical needs of matchmaking events organized by *piranha womex AG*, and variants as well as extensions (some of which are discussed in [2]) may be useful in other contexts.

3 Matchmaking Scheduling in ASP

In this section, we present our ASP encoding of matchmaking scheduling. A matchmaking instance is given by facts as follows: `time(t)` and `location(l)` for each time slot $t \in T$ and location $l \in L$, respectively, `works_for(p,c)` for each $w(p) = c$, `time_pref(p,t)` for each $t \in t(p)$, and `match(c1,c2)` for each $\{c1, c2\} \in M$.

Listing 1 shows our first-order encoding. Following the guess and check methodology of ASP, we first generate a schedule (predicate `mm` provides $S_{\mathcal{C}}$) that assigns at most one match per time slot and location pair in Line 1. The upper bound of the choice rule neatly encodes the feasibility constraint (1).[4] Given that $S_{\mathcal{C}}$ must be a total function, Line 3–4 force `mm` to be left-total (all matches have to be scheduled), and Line 6–9 require it to be functional (a match must not be scheduled at multiple time slots or locations). Line 11–13 encode the feasibility constraints (2) and (3) by associating exactly one employee per company involved in a scheduled match with the time slot and location pair of the match (predicate `mmperson` provides $S_{\mathcal{P}}$).

Line 15–16 implement the overlap objective (O), where an atom `overlap(p,t,n)` expresses that $n > 0$ for $n = \sigma(p,t) - 1$. The **`#minimize`** statement in Line 16 further asserts that the sum over all n in `overlap(p,t,n)` atoms of an answer set ought to be minimal. The time preference objective (P) is encoded in Line 24–25, where `nonpref(p,t,l)` indicates that person p has a match at an unpreferred time slot $t \notin t(p)$, and the corresponding **`#minimize`** statement in Line 25 aims at a minimal number of `nonpref(p,t,l)` atoms in an answer set. Similarly, the objectives (G), (T), and (U) are encoded by the program parts in Line 27–30, 34–35, and 37–38, respectively. Note that we rely on consecutive integers for representing the linearly ordered set T of time slots. Thus, `T+1` in Line 34 refers to the successor time slot $s($`T`$)$ of `T` mentioned in objective (T). Likewise, *gringo*'s built-in comparisons are used in Line 28–29 to check that a company is idle in-between two time slots `T1` and `T2`. Regarding multicriteria optimization, levels provided in the encoding (level "`@5`" in Line 16 for the most important criterion (O) and then gradually decreasing) represent the precedence of objectives. Moreover, Line 40 confines the visible output to instances of predicates `mm` and `mmperson`, which together comprise a schedule S.

[4] Although one may likewise generate a time slot and location pair per match, experiments with such an alternative encoding led to performance degradations.

Listing 1. Matchmaking scheduling encoding

```
1  { mm(C1,C2,T,L) : match(C1,C2) } 1 :- time(T), location(L).    % schedule matches

3  scheduled(C1,C2) :- mm(C1,C2,_,_).                              % all matches must
4  :- match(C1,C2), not scheduled(C1,C2).                          % be scheduled

6  match_at_loc(C1,C2,L)  :- mm(C1,C2,_,L).                        % a match must be
7  match_at_time(C1,C2,T) :- mm(C1,C2,T,_).                        % scheduled once
8  :- match(C1,C2), not { match_at_loc(C1,C2,_) } 1.
9  :- match(C1,C2), not { match_at_time(C1,C2,_) } 1.

11 comp_at_pair(C,T,L) :- mm(C,_,T,L).                             % schedule persons
12 comp_at_pair(C,T,L) :- mm(_,C,T,L).
13 1 { mmperson(P,T,L) : works_for(P,C) } 1 :- comp_at_pair(C,T,L).

15 overlap(P,T,N-1) :- works_for(P,_), time(T), N = { mmperson(P,T,_) }, 1 < N.
16 #minimize[ overlap(P,T,N) = N@5 ].                              % overlap (O)

18 hasoverlap(C) :- overlap(P,_,_), works_for(P,C).
19 matchc(C,MC)  :- MC = { match(C,_), match(_,C) }, company(C). % match count
20 peoplec(C,PC) :- PC = { works_for(_,C) }, company(C).          % people count
21 timec(TC)     :- TC = { time(_) }.                              % time count
22 :- matchc(C,MC), peoplec(C,PC), timec(TC), PC*TC < MC, not hasoverlap(C).

24 nonpref(P,T,L) :- mmperson(P,T,L), not time_pref(P,T).
25 #minimize{ nonpref(P,T,L)@4 }.                                  % time pref. (P)

27 comp_at_time(C,T) :- comp_at_pair(C,T,_).
28 gap(C,T1,T2-T1-1) :- comp_at_time(C,T1), comp_at_time(C,T2), T1+1 < T2,
29                      not comp_at_time(C,T) : time(T) : T1 < T : T < T2.
30 #minimize[ gap(C,T,N) = N@3 ].                                  % gap (G)

32 :- timec(TC), gap(C,_,N), not { comp_at_time(C,_) } TC-N.

34 tablechange(C,T) :- comp_at_pair(C,T,L1), comp_at_pair(C,T+1,L2), L1 != L2.
35 #minimize{ tablechange(C,T)@2 }.                                % table change (T)

37 usedtime(T) :- comp_at_time(_,T).  usedloc(L) :- match_at_loc(_,_,L).
38 #minimize{ usedtime(T)@1, usedloc(L)@1 }.                       % used res. (U)

40 #hide.  #show mm/4.  #show mmperson/3.
```

Additionally, in Line 18–22 and 32, we make some “redundant” domain knowledge explicit, which is not necessary but may improve solving performance. The integrity constraint in Line 22 states that it is impossible to schedule employees of an overbooking company, which participates in more matches than the number of available time slots multiplied by the man power of the company, without overlap. Furthermore, Line 32 expresses that the length of a gap (a sequence of idle time slots) together with a company’s scheduled time slots cannot exceed the total number of available time slots.

4 Experiments

In our experiments, we ran *gringo* (3.0.5) for grounding and *clasp* (2.1.1) for solving. All experiments were performed on a 2.5GHz Intel Core Duo machine with 4GB memory under MacOS X (10.7.5), imposing 3600 seconds as time limit. We configured *clasp* to use the VSIDS decision heuristic (`--heuristic=Vsids`), which was the best

						clasp													
Instance	#m	#t	#l	#c	#p	O	P	O	P	G	O	P	G	T	O	P	G	T	U
2on2	2	2	2	3	3	∗0	0	∗0	0	0	∗0	0	0	0	∗0	0	0	0	3
3on2	3	2	2	3	3	∗1	0	∗1	0	0	∗1	0	0	1	∗1	0	0	1	4
39on14	39	14	20	15	15	∗0	0	∗0	0	0	0	0	22	23	0	0	16	36	31
180on4	180	4	80	100	100	36	0	88	0	69	84	0	43	134	90	0	52	119	80
ffm11li	13	10	26	9	9	∗0	0	∗0	0	0	∗0	0	0	7	∗0	0	0	7	9
ffm11cr	19	10	26	13	16	∗1	0	∗1	0	0	1	0	0	9	1	0	0	8	14
ffm11mu	24	11	26	14	16	∗0	1	∗0	1	0	0	1	0	14	0	1	0	14	11
wmx10	77	8	14	26	51	∗0	0	∗0	0	0	0	0	0	70	0	0	0	72	22
wmx11	69	8	14	26	26	13	3	15	4	1	15	4	13	83	15	5	12	69	22
wmx11e	59	8	14	26	26	0	6	0	6	1	0	6	2	64	0	6	2	70	22
wmx11m	82	8	14	26	26	22	12	22	11	16	22	14	28	89	22	13	20	87	22
wmx11p	69	8	14	26	52	7	32	0	23	25	8	33	34	63	6	29	19	73	22
wmx12	60	7	14	54	54	∗1	0	∗1	0	0	1	0	0	26	1	0	0	30	19
wmx12m	89	7	14	54	54	5	2	5	2	29	8	2	29	81	6	2	36	82	21
wmx12p	60	7	14	54	75	∗0	0	∗0	0	0	0	0	0	23	0	0	0	24	19

Table 1. Benchmark results with gradually increasing hierarchy of objectives

configuration we found so far. We also tried *clasp* options for dedicated multi-criteria optimization [3], but we did not achieve performance improvements with them. Moreover, note that we are unaware of any other freely available matchmaking scheduling system that would be directly comparable with our ASP-based approach.

Table 1 displays benchmark results. The first two columns list instances and their properties, where #m, #t, #l, #c, and #p are numbers of matches, time slots, locations, companies, and persons, respectively. While the first four instances are crafted, the others are real-world instances from Frankfurter Musikmesse 2011 and WOMEX 2010–2012 as well as extended versions of them with additional matches or persons. The columns O, P, G, T, and U show objective values for (O), (P), (G), (T), and (U), respectively, w.r.t. the best answer set found by *clasp* within the allotted time. We started with a relaxed problem using only (O) and (P) as objectives (results shown in the third column) and then gradually added more objectives in the order of precedence, leading to the full problem with all objectives (results shown in the last column). Whenever *clasp* proved a solution to be optimal, the corresponding entry starts with $*$. For instance, the entry "$*1\ 0\ 0$" for the ffm11cr instance indicates that a schedule with one overlap but no unpreferred time slots or gaps has been found and proven to be optimal by *clasp*. Although more objectives add to the difficulty of multi-criteria optimization, the gradual solutions for instances show only slight degradations. For example, objective values for (P) on the wmx11m instance vary between 11 and 14, and the values are not monotonically increasing with the number of objectives. According to *piranha womex AG*, schedules computed with our ASP-based approach are satisfactory and on some real-world instances even better than previous hand-made schedules.

5 Discussion

We presented an ASP-based approach to matchmaking scheduling, which is a highly combinatorial multi-criteria optimization problem. Our approach allows for solving real-world matchmaking instances that were previously dealt with by special-purpose algorithms. The presented ASP methods are used by *piranha womex AG* for computing matchmaking schedules for the world music exposition (WOMEX) and other fairs.

Although matchmaking scheduling can be regarded as a form of timetabling, to our knowledge, it has not yet received much attention from the timetabling community. Common search methods for timetabling include local search [4], constraint programming [5], and satisfiability solving [6]. Moreover, an extension of disjunctive logic programming by soft constraints was proposed for modeling school timetabling [7]. The commercial matchmaking scheduling system *b2match* [8] supports only gap minimization among the various objectives we considered.

Acknowledgments. This work was partially funded by DFG grant SCHA 550/9-1.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
2. Glase, T.: Timetabling with answer set programming. Diploma thesis, Institute for Informatics, University of Potsdam (2012)
3. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-criteria optimization in answer set programming. In ICLP'11, Dagstuhl Publishing (2011) 1–10
4. Cambazard, H., Hebrard, E., O'Sullivan, B., Papadopoulos, A.: Local search and constraint programming for the post enrolment-based course timetabling problem. Annals of Operations Research **194**(1) (2012) 111–135
5. Baptiste, P., Pape, C., Nuijten, W.: Constraint-Based Scheduling. Springer (2001)
6. Achá, R., Nieuwenhuis, R.: Curriculum-based course timetabling with SAT and MaxSAT. Annals of Operations Research (2012) Published online
7. Faber, W., Leone, N., Pfeifer, G.: Representing school timetabling in a disjunctive logic programming language. In WLP'98, (1998) 43–52
8. b2match: `http://www.b2match.com/info/pages/scheduling/`