Events Organizer on Social Networks

Julien Odent
Oct 15, 2012

Events Organizer: Overview

- Problem definition
- Modeling
  - Specifications
  - Constraints
  - LS problem
- Implementation
- Improvements
- Extensions

Events Organizer: Problem definition

Given a set of events and the attendees' preferences, find the best assignment of participants to each event

Events

- occur at a given time
- have a maximal capacity

Participants

- (un)subscribe to events
- specify their preferences

Events Organizer: Modeling

Assume n, the number of participants and m, the number of events

INPUTS:

c: capacity vector of length m

c_i = maximum capacity of event i (0 if unlimited)

p: preferences matrix of size n x m

p_{ij} = 1 if participant i wishes to attend event j

0 otherwise

d: overlapping matrix of size m x m

d_{ij} = 1 if event i overlaps with event j

0 otherwise

Events Organizer: Modeling

Assume n, the number of participants and m, the number of events

OUTPUT:

s: attending matrix of size n x m

s_i = 1 if participant i attends event j

0 otherwise

Events Organizer: Modeling

Max capacity constraint $\sum_i^m s_{ij} \leqslant c_j \; \forall j$

"Non-ubiquity" constraint $\forall k \forall i, j : i < j, s_{ki} = s_{kj} = 1 \Rightarrow d_{ij} = 0$

Events Organizer: LS problem

Objectives

- Maximize the number of attendees $\sum_i^n \sum_j^m s_{ij}$
- Balance the ratio participants / capacity among all events

$$\prod_i^n \frac{\sum_j^m s_{ji}}{c_i}$$

(= 1 when $c_i = 0$)

Neighborhood

- Solutions increasing by one the attendees
- Solutions resulting from swapping two exclusive events

Events Organizer: Implementation

- Copy p into s

  s is consistent w.r.t. the capacity vector (waiting queues)

- Make s consistent with overlapping events (d matrix) by removing the problematic attendees

  => s is the initial solution

- Tabu search (without random restarts)

  Fitness: one of the two objectives (previous slide)

  Features: for now, just the whole solution matrix

  Neighborhood: one of the two solutions sets (previous slide)

Events Organizer: Implementation

- One waiting queue by event containing participants' index
  - FIFO (first-come, first-served)
- When a participant subscribes to an event
  - if the event is not full yet, update the preferences' matrix
  - else append its index into the event's waiting queue
- When a participant unsubscribes to an event
  - if the event was full, take the first one from queue
  - update the preferences' matrix

Events Organizer: Improvements

- Features: use the common rows or/and cols of the best candidate and the best solution
- Neighborhood
  - remember the inconsistent solutions to avoid trying them again
  - explore larger neighborhood
  - shape according to fitness function

Events Organizer: Extensions

- Participant classes: organizers, VIP guests, regular participants

  => “At least 5 organizers must attend that event”

  => “As a participant, I want to attend at most 3 events among...”

  => “As an organizer, I want to prioritize VIP guests”

- Continuous overlapping events

  => “I'll drop by at that event around 8 p.m., then at another one around 9:30 p.m...”

- Min attendees for an event to actually occur

  => Drop a few events in order to maximize the others (threshold)

Events Organizer: Extensions

- Total ordering of the preferences of a given participant
- Waiting queues: setting priority

  => by amount of friends present at a given event

  => “manually” set by organizers
- Event classes: party, sport competition

  => different configuration from class to class