MOPaC: The Multiple Offers Protocol for Multilateral Negotiations with Partial Consensus

Pradeep K. Murukannaiah and Catholijn M. Jonker and W.Pasman

September 27, 2021

1 Introduction

Multilateral negotiations enable group decisions involving two or more parties. The SAOP and AMOP protocols [1] facilitate multilateral negotiations, but require a (full) consensus among parties. However, negotiations requiring full consensus can be too restrictive, e.g., in the following scenarios.

Example 1.1 *Consider a meeting scheduling negotiation among n (> 2) parties. Ideally, we would like all parties to agree on the meeting but the meeting can take place even if a few parties cannot make it. That is, if there are no offers that all parties agree on, but there are offers that a (large) subset of the parties agree on, the negotiation can still succeed with a partial consensus.*

Example 1.2 *Consider the process of coalition building in a multi-party political system [2]. In an election, n (> 2) parties contested, and a certain number of candidates won from each party. However, none of the parties have a majority, on their own. Thus, the parties must negotiate and build a coalition to form the government. A partial consensus is sufficient to build a successful coalition.*

Example 1.3 *Consider that a group of n (> 2) friends would like to share (rent) flats with each other during their visit to a city. Each person in the group has preferences on the characteristics of the flat he or she wants to rent and the characteristics of their flatmates. The friends engage in a negotiation to find suitable flatmates. This negotiation can yield more than one successful deal each involving a partial consensus among a subset of the friends.*

This report contains two parts:

1. A formal description of the protocol (section 2+3).
2. A description of the implementation (section 4).

2 The MOPaC Protocol

Let A be a set of n agents in a multilateral negotiation. $\forall i, A_i : 0 < i \leq n$, A_i, identifies one of the agents, where $\forall i, j : 0 < i, j \leq n$ and $i \neq j \implies A_i \neq A_j$.

Each agent A_i has a power $p_i \in \mathbb{N}$ in the negotiation. Thus, the power, p, of a full consensus among agents is:

$$p_{max} = \sum_{i=1}^{n} p_i \tag{1}$$

Let p_{min} be the minimum power required for any (partial) consensus to form in this negotiation, which is sent as a parameter to the protocol.

2.1 Protocol

A multilateral negotiation with the MOPaC protocol runs for one or more rounds. Each round consists of the following phases.

Bidding phase: Let B be a bidding space of m possible bids. Each agent puts a bid on the table. Let $b_{ij} \in B$ be agent A_i's bid in round j.

After the bidding phase, each agent is communicated the bid and power of each agent. That is, the following list is communicated to each agent:

$$\Big[[A_1, b_{1j}, p_1], \ \ldots [A_n, b_{nj}, p_n]\Big] \tag{2}$$

Voting phase: Each agent votes (*accept*/*reject*) on each bid on the table. With each *accept* vote, an agent must indicate:

(1) a *minimum consensus threshold*, C_{min}: $p_{min} \leq C_{min} \leq p_{max} \in \mathbb{N}$, and
(2) a *maximum consensus threshold*, C_{max}, $C_{min} \leq C_{max} \leq p_{max} \in \mathbb{N}$.

That is, the two possible votes an agent can provide on a bid b_i (including the agent's own bid) are:

$$v_i = \begin{cases} \langle b_i, \text{accept}, C_{min}, C_{max} \rangle, \text{or} \\ \langle b_i, \text{reject} \rangle. \end{cases} \tag{3}$$

By accepting a bid b_i with $C_{min} = m$ and $C_{max} = n$, an agent indicates that it will accept the bid b_i if a partial consensus group of *power* at least m and at most n forms around bid b_i (Section 3).

After the voting phase, each agent is communicated the votes of each agent. That is, the following list is communicated to each agent:

$$\Big[[A_1, \langle b_1, v_1 \rangle, \ \ldots \langle b_m, v_m \rangle], \ \ldots [A_n, \langle b_1, v_1 \rangle, \ \ldots \langle b_m, v_m \rangle]\Big] \tag{4}$$

Opt-in phase: Each agent votes again, as before, but with constraints that an agent cannot *reject* or reduce the C_{min} value of a bid it had *accept*ed in the voting phase. That is, if an agent accepted b_i with vote v_i in the voting phase, its vote for b_i in the opt-in phase v'_i should be as follows.

$$\begin{aligned} \forall v_i : v_i &= \langle b_i, \text{accept}, C_{min}, C_{max} \rangle, \\ v'_i &= \langle b_i, \text{accept}, C_{min} \leq C'_{min} \leq p_{max}, C_{min} \leq C'_{max} \leq p_{max} \rangle. \end{aligned} \tag{5}$$

However, an agent can *accept* a bid it had *reject*ed in the voting phase.

$$\begin{aligned} &\forall v_i : v_i = \langle b_i, \text{reject} \rangle, \\ &\quad v'_i = \begin{cases} \langle b_i, \text{reject} \rangle, \text{or} \\ \langle b_i, \text{accept}, p_{min} \leq C'_{min} \leq p_{max}, C'_{min} \leq C'_{max} \leq p_{max} \rangle. \end{cases} \end{aligned} \tag{6}$$

Continuation or termination: There is more than one way a MOPaC negotiation can terminate. The following are two potential options. In both of these options, the continuation or termination decision is made after computing the viable consensus groups at the end of the second round of voting.

(1) Determine a viable consensus group with the largest power (see Section 3). If there is a tie for the group with largest power, break the ties randomly. For the agents in the chosen viable consensus group, the negotiation terminates with a deal. For the remaining agents, the negotiation terminates without a deal. This kind of termination is ideal, e.g., when negotiating to schedule a meeting (Example 1.1) or form a government (Example 1.2).

(2) Determine a viable consensus group with the largest power (see Section 3). If there is a tie for the group with largest power, break the ties randomly. Determine if there are more viable groups consisting of the remaining agents. If so, the negotiation terminates with a deal for such agents, too. The remaining agents go to the next round of negotiation. The negotiation continues until the deadline or until one or no agent remains in this round, whichever happens first. This type of termination is ideal, e.g., when negotiation for flatmates (Example 1.3).

3 Partial Consensus Formation

Given a set of agents $C = \{A^C_1, \ldots A^C_m\}$, who each voted *accept* for a bid b_i, C is a partial consensus group on b_i. The **power** of the consensus C is:

$$p_C = \sum_{A_i \in C} p_i. \tag{7}$$

A consensus C for a bid b_i is **viable** iff $\forall A_i \in C$, $C_{min} \leq p_C \leq C_{max}$. That is, a viable consensus group for a bid b_i consists of a set of agents that accepted b_i and the power of the group is within the minimum and maximum consensus thresholds indicated by each agent in the group.

3.1 Computing Viable Consensus Groups

The following is a naive approach for computing all viable consensus groups at the end of a MOPaC round. Although naive, this approach should be computationally feasible for a small number (e.g., $n \leq 10$) of agents.

1. Enumerate all bids b_i in a round.
2. Compute the set of all possible agent groups C_P^A such that a group of agents $c_i \in C^A$ iff $c_i \in P^A$, the power set of A and $|c_i| > 1$.

   $|C^A| = 2^n - n - 1$.
3. For each bid b_i and agent group c_i, determine if c_i is (1) a consensus group, and (2) viable. If so, add $\langle b_i, c_i \rangle$ to the list of viable groups, and proceed.

Optimization note: The apriori algorithm used in association analysis [3] can be used to prune the search space. That is, if a set c_i is not a consensus group, none of its subsets will form a consensus group.

4 Implementation

An implementation of the formal protocol is available in GeniusWeb. The details differ from the formal definition, because it is more detailed, restricted and is object oriented rather than mathematical.

This section gives an overview of the classes involved with MOPaC. For all the fine details, please refer to the GeniusWeb javadoc and wiki pages.

4.1 Protocol

At the top level, the MOPaC protocol is implemented by the `MOPAC` class. The state of the protocol is defined in the `MOPACState`, which can be seriailized, eg to a log file to log the final outcome details. The initial `MOPACState` is defined through the `MOPACSettings` object.

The `MOPACSettings` object contains a `Deadline`, a List of `TeamInfo` objects and a `VotingEvaluator`.

The `Deadline` is a mechanism to ensure termination. It does this by placing an absolute time limit, plus possibly a number-of-rounds limit, on the protocol. If one of the limits is exceeded, the negotiation will just stop, and all parties that did not reach a deal leave the negotiation without a deal.

The implementation is very specific about handling termination, both of parties doing an action not allowed by the protocol, when it ends a negotiation (walks away) and when the deadline is reached. If a party has placed a `Vote` or opted in, that vote or opt-in remains valid even after it terminates.

The `TeamInfo` is in the MOPaC case just a single `Party` plus its `Profile` and its `Parameters`. The Parameters is where the power of each party will be set, with key "power" and an integer as value. Parameters are optional, power=1 if not set. There may be other, party-specific parameters present.

The protocol keeps track of the phase using the `OfferPhase` (parties are expected to bid), `VotingPhase` (parties are expected to vote) and `OptInPhase` (parties are expected to opt in).

The `VotingEvaluator` is an implementation of the Partial Consensus Formation procedure of section 3. However it is much more detailed and has a few variants. `VotingEvaluator` basically is a function that takes a set of `CollectedVotes` and can determine the `Agreements`. There are two `VotingEvaluator` types:

- `LargestAgreement`: the negotiation ends as soon as a consensus group can be formed, with the consensus that has the largest total power as outcome.
- `LargestAgreementsLoop`: if a consensus group can be formed, create the consensus group with the largest total power. Remove the parties in the consensus group from the negotiation. If parties remain, repeat. If less than 2 parties remain, the negotiation ends. Otherwise, negotiation continues with the remaining parties.

4.2 The Party's side objects

A negotiation party usually sees very little of all the protocol details: it just waits for a message from the protocol on what is expected next, and then acts accordingly. These messages are communicated through a number of objects:

`SessionSettings`: The usual information to start up a session, containing the party's `Parameters`, `Profile` etc.

`YourTurn`: the party receives this object to indicate the start of the Bidding phase of the protocol. Normally the party responds with an `Offer`

`Voting`: the party receives this when it's time to vote on the bids on the table.Normally the party responds with a `Votes` containing every `~Vote` that it can accept.

`OptIn`. The party receives this when it's time to opt in. The party again normally responds with a `Votes` object with a number of `Vote` objects. If a `Vote` was already voted for in the previous phase, the new Vote must extend the old Vote: iit must accept at least the same as the previous phase, to ensure that parties can not back off from a deal they already voted for.

In all phases, a party can decide to `EndNegotiatino`. This basically means the party walks away. Remaining parties just continue negotiating.

References

[1] Reyhan Aydoğan, David Festen, Koen V. Hindriks, and Catholijn M. Jonker. Alternating offers protocols for multilateral negotiation. In Katsuhide Fujita, Quan Bai, Takayuki Ito, Minjie Zhang, Fenghui Ren, Reyhan Aydoğan, and Rafik Hadfi, editors, *Modern Approaches to Agent-based Complex Automated Negotiation*, pages 153–167. Springer International Publishing, Cham, 2017.

[2] Christophe Dupont. Negotiation as coalition building. *International Negotiation*, 1(1):47 – 64, 01 Jan. 1996.

[3] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson, Boston, 2006.