開催時間を譲渡する会議日程調整法の実装と評価

小峰 和也 †　平井 和則 ‡　中村 勝一 ‡　程 子学 †
† 会津大学コンピュータ理工学部 ‡ 会津大学コンピュータ理工学研究科
〒 965-8580 福島県会津若松市一箕町鶴賀
email:{s1031027,m5021217,m5021214,z-cheng}@u-aizu.ac.jp

今までの会議日程調整は, ある参加者が参加可能な複数の会議のうち, どの会議にも参加できない場合の時間調整に有効な交渉プロトコルを与えていない. 本稿では, 複数の会議で同じ参加者が必要とされる場合, 会議間で開催可能時間の譲り合いによって会議が開催できるような日程調整を考え, そのための交渉プロトコルを与える. そして, 提案する方法の有効性と効率を検証するため, Java 言語を用いて実験システムを試作した. また, 実験の結果として,*depth* というパラメータを用いることで時間満足度を, *length* というパラメータを用いることでメッセージ数を調節が可能である.

Implementation and Evaluation of the Meeting Scheduling Method based on Concession of Available Time

Kazuya Komine† Kazunori Hirai‡
Shouichi Nakamura‡ Zixue Cheng†
†Department of Computer Software, The University of Aizu
‡Graduate School of Computer Science and Engineering, The University of Aizu
Aizu-Wakamatsu City, Fukushima 965-8580 Japan
email:{s1031027,m5021217,m5021214,z-cheng}@u-aizu.ac.jp

Most of meeting scheduling methods can't arrange effective schedules, in the case where more than one meeting require some common participants, and they can't held without the attendance of the participants. In our research, we consider a meeting scheduling method, by which more conflicting meetings can't be held by concession of time among meetings in that case, and we propose a negotiation protocol for this scheduling. In order to verify the effect and efficiency of the method, we have implemented an experiment system. Our experiment shows using parameter "*depth*", meeting available times are satisfied by users, and using parameter "*length*", message complexity can be controlled.

1 はじめに

高度情報化社会により, コンピュータやネットワークが発展し, 分散環境を利用した集団の共同作業が可能となり, それを支援するグループウェアの研究・開発されている. グループウェアの主要な研究として, ソフトウェアの共同開発, ドキュメントデータベース, ワークフロー管理, スケジュール調整などが挙げられるが, 本研究では, スケジュール調整の一つである会議日程調整について考える.

会議日程調整に関する研究は盛んに行なわれており, 多くの会議日程調整システムが開発, 市販化されている [1]. これらのシステムの多くは電子カレンダーや掲示版を用いて, 開催可能な時間を掲示したり, 会議時間を確認するためのものであり, 利用者の時間に対する優先順位やカレンダー上の利用可能な時間を考慮し, 他の参加者やそのエージェントとの交渉を自律的に行うといった, 柔軟な日程調整を行なうことが出来ない.

[2],[3] では,3 つの調整法を提案し, 他の参加者に不利な操作を不可能にしているが, 参加者に参加の義務が強いられるシステムのみに有効であったり, [4] では, 他の参加者やそのエージェントと会話ができる Active Email を用いた調整法を提案しているが, 日程の決定, 変更の承認をユーザーに求めるため, エージェントによる完全な自律的調整はされない. また, [5] では, 分散型の会議日程調整プロトコルを提案し, プライバシー情報を他のエージェントや人に知られないように調整を行なうが, 同時に開催できない会議対して, 合意を得るまで単純な取消や再確認を繰り返し送信するので, メッセージ複雑度が大きくなってしまう. [6] では, 複数の会議間の日程調整を行なうが, 広い範囲での会議間の交渉をする調整を行なうことはできない.

本研究は, 複数の人が複数の会議に参加する状況も想定し, 会議の各参加者の時間に対する優先度を考慮した日程調整を考える. その際, 会議間の開催可能時間に関する譲り合い交渉プロトコルを与え, そのプロトコルは, 会議間で開催時間を譲り合うことで, 同じ参加者が必要とされる複数の会議間で, それぞれの会議の開催時間を矛盾なく調整を行なうことができる. 譲渡交渉プロトコルの基本的方針は, グラフ理論における増大道を発展させ, 複数の会議と複数の時間における増大道を会議エージェントと参加者エージェント間の通信により発見し, 譲渡交渉を行なうものである. また, 他の会議に譲渡要求を出すことが可能な範囲を表すパラメタと他の会議からの譲渡要求に応える範囲を表すパラメタを導入し, より多くの会議を開催するという目標と譲渡を行なう際に生じるコストや他の会議へ与える影響などのバランスを考慮する交渉を行なう.

本論文の構成は, 2 でモデルと問題定義を示し, 3 では譲り合いの基本的な考え方及び日程調整法の提案, 4 では 3 で提案した日程調整法を実装し実験結果を述べ, 最後に 5 で, 本論文をまとめる.

2 モデルと問題定義

2.1 モデル

各会議と各参加者にそれぞれ 1 つのエージェントを設け, そのエージェント同士の交渉により会議日程を調整する. $C = \{c_1, .., c_i, .., c_n\}$ は, 会議エージェントの集合を表し, $M = \{m_1, .., m_l, .., m_m\}$ は, 参加者エージェントの集合を表している. そして, 各会議 c_i は, その会議に参加可能な参加者エージェントの集合 ($M_i \subseteq M$) を所持している. また, 各参加者 m_l は, すべての参加可能な会議について会議エージェントの集合 ($C_l \subseteq C$) を所持している.

各参加者エージェント m_l は, 会議に出席可能な空き時間とその出席可能な空き時間に対する m_l が会議を開催して貰いたい時間順に優先度を設けその集合 AT_l を持つ.

$$
\begin{aligned}
AT_l &= \{t_p, pr(t_p)\} \\
t_p &\ ; \ m_l\text{が出席可能な時間} \\
pr(t_p) &\ ; \ \text{時間 } t_p\text{に対する優先度}
\end{aligned}
$$

ただし, ある時間 t_p を参加可能時間として持つ参加者の総数が会議開催に必要な最低人数 q_i に満たない場合は, その t_p を GT_i から除くものとする. 本論文での優先度の定義として, $pr(t_p)$ の値が小さい正整数ものほど優先度が高いものとする.

各会議エージェント c_i は, 開催のための最低出席人数 $q_i (0 < q_i \leq |M_i|)$ を設る. また,c_i は各参加者の出席可能な時間とその時間に対する優先度の和集合 GT_i を持つ.

$$
\begin{aligned}
GT_i &= \{t_p | t_p \in \bigcup_{\forall m_l \in M_i} \} \\
AT_l &\ ; \ t_p\text{が } q_i\text{人以上含むものに限る}
\end{aligned}
$$

図 1 は 3 つの会議と 4 人の参加者間においてすべての会議が開催可能な例を示している." ●" は会議エージェント," ○" は参加者エージェントを表している. ある会議とその会議に参加可能な参加者は, 会議エージェントと参加者エージェントを結ぶ線で表している.

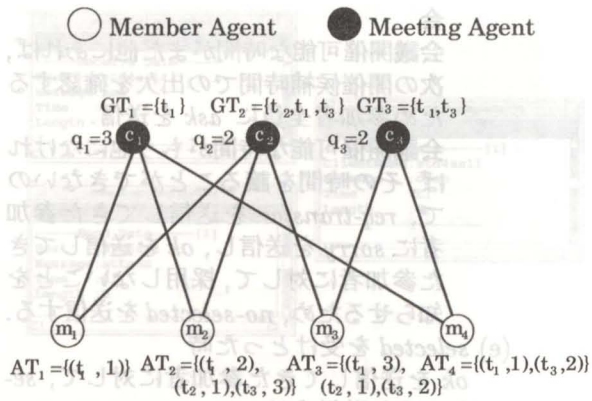

図 1: モデル例

2.2 問題定義

本論文での会議日程調整とは,2.1 で定義したモデルの上で以下の条件を保証し, 各参加者がいつ, どの会議に出席するかを決定することである.

1. q_i 人以上の出席がないと会議 c_i は開催できない（最低人数出席条件）.
2. ある参加者 m_l は, 同時に複数の会議に出席できない（同時参加不可能条件）.
3. 共通の参加者の出席を必要とする複数の会議が同時に開催できない場合, より多くの会議を開催させるため, 一部の会議の開催時間の譲渡により日程調整の交渉を行なう. その譲り合いは, 会議エージェントと参加者エージェントの交渉により決定する.

3 会議日程調整法

3.1 基本的考え方

共通の参加者の出席を必要とする複数の会議が同時に開催できない場合, ある会議の開催予定時間を変更（譲渡）することにより日程の調整を行なう. また, この交渉は開催時間の譲渡を希望する会議エージェントが, 共通な参加者をもつ別の会議に対し, その参加者エージェントを通して開催予定時間の変更を要求することにより行なう.

譲渡交渉は, 会議と時間の 2 部グラフにおける増大道のサーチの考え方に基づいて行なう.[7]

また, 他の会議に開催時間を譲り, 最も開催条件の悪い時間まで開催可能時間を変更することを防ぐために, "*depth*" という変数 (正整数) を用い, ある増大道が見つかるまで, サーチし続けることを防ぐために, "*length*" という変数 (正整数) を用いる. 前者は, 優先度の高い開催可能時間から最大 *depth* 番目の開催可能時間まで譲ることは可能で, それ以上は譲ることは出来ないことを示し, 会議の満足度を向上することができ, 後者は, 交渉可能な会議の範囲を示し, 行き過ぎた交渉を防ぐことが出来る.

送り主	メッセージ	意味
会議	request	AT を要求
	open	会議を開催
	no-open	会議開催不可
	free	出席自由選択
	ask	出席可能確認
参加者	answer	AT を送信した返答
	attend	会議に出席
	absent	会議に欠席
	overlapped	他の会議の時間と重複
共通	sorry	譲渡不可能
	ok	譲渡可能
	selected	譲渡の選択
	req-transfar	時間の譲渡を要求
	no-selected	譲渡の不必要
	recognize	作業の終了確認

表 1: メッセージリスト

3.2 日程譲渡プロトコル

本研究の日程調整で使用されるメッセージは以下の表 1 で示した通りである.

本研究の日程調整は, 新しく提示（生成）された会議 c_i に対して日程調整を行なう. 同時に生成された会議に対しては相互排他的にどれか 1 つの会議について日程調整を行ない, 調整終了後に調整権を他の会議に移行する.

日程調整が行なわれる前提として, 全ての会議について, 参加者, 最低出席人数, *depth*, *length*, の情報が会議主催者によって設定されているものとする. また, 全ての参加者は, 出席可能時間と, その時間に対する優先度の情報をあらかじめ設定し, ある時間を譲渡可能であるかどうかの初期状態は自動的に全て true（true であれば譲渡可能である）に設定されているものとする.

会議エージェント動作

日程調整が行なわれる会議 c_i における作業

1. **譲渡前初期設定**
   (a) 会議は, 参加権を持つ全ての参加者に対して時間の優先度を要求するため *request* メッセージを送信.
   (b) 会議に参加権を持つ全ての参加者から *answer* を受けとった後, 時間優先度の高い順にソートし, *ask* を送信.
   (c) 会議に参加権を持つ全ての参加者から *attend*, *absent*, *overlapped* のうちのい

づれかを受けとった場合, *attend* の人数が最低開催人数を越えていれば *attend* を送信してきた参加者に対し *open* を それ以外の参加者には *free* を送信する.
越えていなく,且つ *attend* の人数と *overlapped* の人数の和が最低開催人数を越えていて,*length* が 1 以上の時, *overlapped* を送信してきた参加者に *req-transfar* を送信.
どちらでもなければ, *no-open* を全員に送信.

2. 譲渡処理

(a) 譲渡要求 *req-transfar* が来たら, 会議は参加権を持つ全ての参加者に対して次の開催候補時間があり, *depth* が 0 以上の時, 次の開催候補時間での出欠を確認するため *ask* を送信.

(b) 次の開催候補時間での開催が可能であれば *ok* を, そうでなければ *overlapped* を送信してきた参加者に *req-transfar* を送信. もし, *overlapped* を送信してきた参加者が存在しなかった場合, 譲渡不可能なため *sorry* を *req-transfar* を送信してきた参加者に送信.

(c) *req-transfar* を送信した参加者分だけ *ok*, *sorry* を受けとり, *ok* と *attend* の和が最低開催人数を越えていた時
最低開催人数になるまで *ok* を採用し, 越えたら採用した方には *selected* を, 採用しなかった方には *no-selected* を *ok* 送信してきた参加者に送信する.

i. 日程調整権を所持している会議の場合
attend を送信してきた参加者には *open* を, *absent*, *sorry* を送信してきた参加者に対して出席する必要がないことを知らせるため, *free* をそれぞれ送信する.

ii. 日程調整権を所持していない会議の場合
req-transfar を送信してきた参加者に対し時間の譲渡が可能なことを知らせるため, *ok* を送信する.

(d) *req-transfar* を送信した参加者分だけ *ok*, *sorry* を受けとり, *ok* と *attend* の和が最低開催人数を越えていなかった時

i. 日程調整権を所持している会議の場合
会議開催が不可能であるということを知らせるため, 参加者全員に *no-open* を送信し, *ok* を送信してきた参加者に対して, 採用しないことを知らせるため, *no-selected* を送信し, 日程調整権を次の会議に移行する.

ii. 日程調整権を所持していない会議の場合
会議開催可能な時間がまだ他にあれば, 次の開催候補時間での出欠を確認するため参加者全員に *ask* を送信.
会議開催可能な時間がもう他になければ, その時間を譲ることができないので, *req-transfar* を送信してきた参加者に *sorry* を送信し, *ok* を送信してきた参加者に対して, 採用しないことを知らせるため, *no-selected* を送信する.

(e) *selected* を受けとった時
ok を送信してきた参加者に対して, *selected* を送信する.

(f) *no-selected* を受けとった時
ok を送信してきた参加者に対して, *no-selected* を送信する.

(g) *length depth* の増減

i. *ask* を送信した時 *depth*, *req-transfar* を送信した時 *length* をそれぞれ 1 減らす.

ii. *sorry* を送信した時, 状態が戻るため *length* を 1 増加させる.

3. 譲渡後処理
参加者数分の *recognize* を受けとった時

(a) 日程調整権を所持している会議の場合
日程調整権を次の会議に移行する.

(b) 日程調整権を所持していない会議の場合
selected を送信してきた参加者に *recognize* を送信する.

参加者エージェント動作

日程調整が行なわれる会議 c_i に参加している参加者 m_l における作業

1. 会議からの要求に対する返答

(a) 会議からの *request* 要求に対する *answer* の返答.

(b) 会議からの *ask* による開催時間の要求に対する *attend*, *overlapped*, *absent* の返答.

(c) 会議からの *open*, *free*, *no-open* に対する *recognize* 返答

2. ある会議から別の会議へのメッセージの転送

(a) *req-transfar*, *selected*, *no-selected*, を受信した時
譲渡要求先の会議に *req-transfar*, *selected*, または *no-selected*, をそれぞれ転送する.
ただし, ある時間を譲渡中に他の会議からその時間を要求された時, すでにその時間は譲渡中なので, その時間を譲ることができないことを知らせるため, *req-*

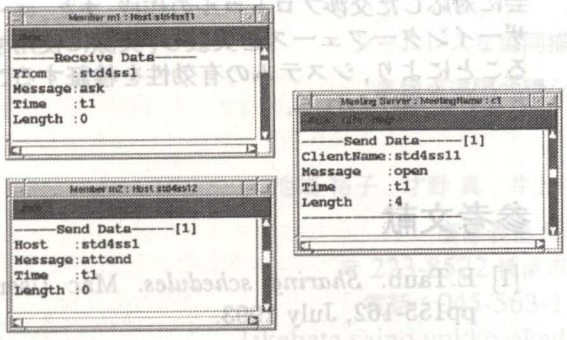

図 2: 実験の実行画面

transfar を送信してきた会議に *sorry* を送信.

(b) *ok, sorry, recognize* を受信した時 *req-transfar* を送信してきた会議に *ok, sorry, recognize* をそれぞれ転送する.

4 実験と評価

本日程調整プロトコルの評価を行なうために, Java 言語を用いて評価実験用システムを構築した. 会議エージェントをサーバ, 参加者エージェントをクライアントとして実装した. 本節では, この実験システムによる評価実験とその結果について述べ, プロトコルの評価を行なう.

4.1 実験方法

実験の方法として, 以下の方法で行なう.

1. 会議の発生順番を入れ換えて, 総メッセージ数, 譲渡の要求数, 譲渡の拒否数, 譲渡の受理数を調べる. 実験例として, 会議はそれぞれ 3 人ずつの参加者を持ち, それぞれ 2 つの会議間において, 共通の参加者は 1 人としたモデルを示す (図 3). 図 3 では, 便宜のため, 集合 GT_i における時間集合状態は, 左から優先順位の高いものとする.
2. *length, depth* と会議の開催数を調べるために, 1. と同じモデル (図 3) を用い, i) c_4 の *depth* の値を 3 ～ 0 に変化させ, 他の会議の *depth* の値は固定する方法と, ii) 全ての会議 *length* の値を 3 ～ 0 に変化させた.

4.2 実験結果と評価

4.2.1 会議の発生順序の入れ換え

実験結果は表 2 に示す. 実験結果より, 開催可能時間が多い会議から調整を開始すると, 以

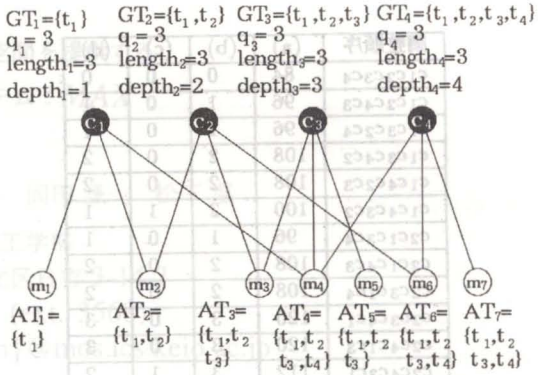

図 3: 実験例

下のことが言える.

- 譲渡要求に対する譲渡拒否が起こる割合が増加する傾向がみられる.
- 総 message 数が増加する傾向がみられる.

4.2.2 depth と会議開催数

表 3 より,*depth* の値を小さくすることによって, その会議の開催時間を過度に下げることを防いで, 調整を行なうことが出来るが, 全体の会議開催数も制限されてしまう.

4.2.3 length と会議開催数

表 4 より,*length* の値を小さくすることにより, メッセージ数を大幅に減少することが出来るが, 全体の会議開催数も制限されてしまう.

5 まとめ

本研究では, 複数の会議で同じ参加者が必要とされる場合, 会議間で開催可能時間の譲り合いによって会議が開催できるような日程調整を考え, そのための交渉プロトコルを与えた. そして, 案する方法の有効性と効率を検証するため, Java 言語を用いて実験システムを試作し, 実験の結果として,*depth* というパラメータを用いることで時間満足度を, *length* というパラメータを用いることでメッセージ数を調節が可能であることを示した. また, 開催可能時間の候補を先に下げることで, その会議が行なう譲渡は 1 回のみとなった. しかし, 必ずしも譲渡が成功するとは限らず, 失敗することによって無駄が生じ, 下げる前の開催可能時間の候補の中に, 譲渡が成功する場合も考えられる. よって, 今回の結果との比較の必要性が挙げられる.

将来の課題として, 必ず会議に参加しなければならない人が存在する場合も考慮し, より実社

調整順序	(a)	(b)	(c)	(d)
$c_1c_2c_3c_4$	84	0	0	0
$c_1c_2c_4c_3$	96	1	0	1
$c_1c_3c_2c_4$	96	1	0	1
$c_1c_3c_4c_2$	108	2	0	2
$c_1c_4c_2c_3$	108	2	0	2
$c_1c_4c_3c_2$	100	2	1	1
$c_2c_1c_3c_4$	96	1	0	1
$c_2c_1c_4c_3$	108	2	0	2
$c_2c_3c_1c_4$	108	2	0	2
$c_2c_3c_4c_1$	120	3	0	3
$c_2c_4c_1c_3$	120	3	0	3
$c_2c_4c_3c_1$	112	3	1	2
$c_3c_1c_2c_4$	90	2	0	2
$c_3c_1c_4c_2$	90	2	0	2
$c_3c_2c_1c_4$	100	2	1	1
$c_3c_2c_4c_1$	112	3	1	2
$c_3c_4c_1c_2$	130	4	0	4
$c_3c_4c_2c_1$	124	4	1	3
$c_4c_1c_2c_3$	120	3	0	3
$c_4c_1c_3c_2$	112	3	1	2
$c_4c_2c_1c_3$	112	3	1	2
$c_4c_2c_3c_1$	104	3	2	1
$c_4c_3c_1c_2$	124	4	1	3
$c_4c_3c_2c_1$	116	4	2	2

(a)—総 message 数 (b)—譲渡要求数
(c)—譲渡拒否数　(d)—譲渡成功数

表 2: 実験結果

depth	(a)	(b)	(c)	(d)	(e)	(f)
3	116	4	2	2	4	t4
2	102	4	3	1	3	t1
1	92	3	2	1	3	t1
0	82	2	1	1	3	t1

(a)—総 message 数 (b)—譲渡要求数
(c)—譲渡拒否数　(d)—譲渡成功数
(e)—開催会議数 (f)—c_4 の開催時間

表 3: 実験結果 (depth)

length	(a)	(b)	(c)	(d)	(e)
3	116	4	2	2	4
2	116	4	2	2	4
1	68	2	2	0	2
0	48	0	0	0	2

(a)—総 message 数 (b)—譲渡要求数
(c)—譲渡拒否数　(d)—譲渡成功数
(e)—開催会議数

表 4: 実験結果 (length)

会に対応した交渉プロトコルの作成. また, ユーザーインターフェースを実装し, 実際に使用することにより, システムの有効性を検証することなどが挙げられる.

参考文献

[1] E.Taub. *Sharing schedules.* Mac User, pp155-162, July 1993.

[2] Eithan Ephrati, Giad Zlotkin, Jeffrey S. Rosenschein. *A non-manipulable meeting scheduling system.* In 13th International Workshop on Distributed Artificial Intelligence, 1994.

[3] Eithan Ephrati, Giad Zlotkin, Jeffrey S. Rosenschein. *Meet Your Destiny: A Non-manipulable Meeting Scheduler.* CSCW, Proceedings of the Conference on Computer Supported Cooperative Work, pp359-371. 1994.

[4] Yaron Goldberg, marilyn Safran, Ehud Shapiro. *Active Mail—A Framework for Implementing Group-ware.* ACM 1992 Conference on Computer Supported Cooperative Work, CSCW, 1992.

[5] Leonardo Garrido-Luna, Katia Sycara. *Towards a Totally Distributed Meeting Scheduling System* Lecture Notes in Artificial Intelligence Vol.1137, pp85-97, 1996.

[6] Sandip Sen, Edmund H. Durfee. *A Formal Study of Distributed Meeting Scheduling: Preliminary results.* In ACM Conference on Organizational Computing Systems, 1991.

[7] 平井和則, 村上隆生, 程 子学. 連続譲渡法に基づいた会議日程調整のための分散アルゴリズムの提案. グループウェア '９６シンポジュウム論文集, pp63-68, 情報処理学会, 1996.

[8] *JAVA プログラミング for the Internet.* MICHAEL D. THOMAS, PRATIK R. PATEL, ALAN D. HUDSON, DONALD A. BALL JR. 毎日コミュニケーションズ, 1997.