3B4-01

マルチエージェントカレンダーシステムによるイベント日程スケジューリングについて

Calendar Event Scheduling for Multi Agent Calender System

磯村厚誌[*1] 大囿忠親[*2] 新谷虎松[*2]
Atsushi Isomura Tadachika Ozono Toramatsu Shintani

[*1]名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology

[*2]名古屋工業大学工学部知能情報システム学科
Dept. of Intelligence and Computer Science, Nagoya Institute of TechnologyJapanese Society for Artificial Intelligence

Making calendar events often need rescheduling of existing one. A solution of the scheduling should satisfy the preferences of all participants of the new events. However, participants would hold private events in parsonal calendar.In theis paper, we present a multi-agent calendar system and formalize a calendar scheduling as a Distrebuted Valued Constraint Satisfaction Problem(DVCSP). In order to solve this DVCSP, we propose a method based on a greedy repair distributed algorithm. Our method allows eache agent to make all available remote copies, and then effective solution is obtained. In our calendar system, users can schedule effectively and satisfy many constraints which contain private information.

1. はじめに

近年普及しているカレンダーソフトの多くは，ユーザ間のコミュニケーションを可能にする機能を有している．しかし，それらは単にコミュニケーションの方法を提供しているだけである．本稿では，複雑なスケジュール調整を半自動的に行うマルチエージェントカレンダーシステムを提案する．この機能は，カレンダーのスケジューリング問題を分散重み付き制約充足問題 (DVCSP) として定式化することで実現する．ユーザには通常のカレンダー情報に対し，制約に関する情報を付加することのみが要求される．本システムはカレンダー上のイベントを変数とし，与えられた制約の情報をもとに自動的にカレンダースケジューリングを行う．また，本システムにおけるカレンダースケジューリングは，基本的に新なイベントが発生した場合の再スケジューリングであると考える．カレンダーには年間，月間を見通して企画されるイベント群や，臨時に実施が必要となるイベントなど様々であり，これらのイベントはある時点で一斉にスケジューリングすることがほぼ不可能である．そのため，あらたなイベントが企画される度に再スケジューリングを行うものとした．この時，再スケジューリングの結果は，与えられた全ての制約を考慮するのみでなく，そのスケジューリングが行われる前の状態を出来るだけ維持している必要がある．この性質は解の安定性と呼ばれ，動的制約充足問題 (Dynamic Constraint Satisfaction Problem : Dynamic CSP) の基本的な考え方である．本システムでは，解の安定性を実現するために，変更したくないイベントをその度合いに応じて制約として記述する．これによって，ユーザの意思を反映した再スケジューリングが可能となる．また，DVCSP として定式化することにより，プライベートなイベントデータの保護と，それに関するユーザの意向の反映を同時に実現する．

連絡先: 磯村厚誌，名古屋工業大学大学院知能情報システム工学科新谷研究室，電話 052-744-3153，FAX 052-735-5477，isomura@ics.nitech.ac.jp

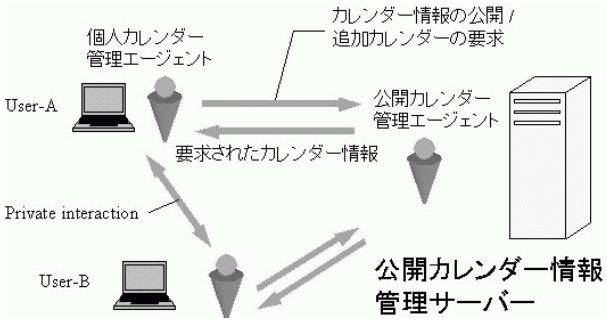


図 1: システム概要

2. 分散カレンダーシステム

2.1 システム概要

本システムは，図 1 に示すように，公開されたカレンダー情報の管理を行なう公開カレンダーサーバーと，各個人のカレンダーシステムで構成される．それぞれ，公開カレンダー管理エージェント (サーバーエージェント) と個人カレンダー管理エージェント (個人エージェント) によって操作される．個人エージェントはサーバーエージェントに対しカレンダー情報を公開し，サーバーエージェントは，公開された全てのカレンダー情報から各個人エージェントの要求に対応したカレンダーを編集，提供する．また，各個人エージェント間での通信も可能であり，複数の参加者を持つイベントについて非公開で計画を立てることが可能である．

カレンダーイベントのスケジューリングを行う際には，サーバエージェントはスケジューリング対象となる各個人エージェントどうしのネットワークを確立することが仕事であり，その後は個人エージェント間のみでスケジューリングに関する操作が行われる．そしてスケジューリング後の公開カレンダー情報は再び個人エージェントからサーバエージェントに向けて更新される．

2.2 カレンダーシステムの応用事例

本研究の具体的な応用として，研究に関するミーティング日程スケジューリングがある．この場合締め切りや開催日などの日程に関する情報は，全てのユーザにとって他のスケジュールに対する制約となり得るが，この制約の情報は，気づいたユーザがカレンダーに公開することで，サーバエージェント側から全てのエージェントに向けて制約として伝えられる．日程に関する情報については，ホームページなどの情報源が特定できる場合には，サーバーエージェントに情報を収集させることも可能である．また，CSP が適用されている他の問題にも，イベントの変域や制約をある一定の規則に従わせることで適応可能なものがある．その例としては，会議日程スケジューリング [1] やナーススケジューリング [2] などが挙げられる．これらの実例に対する本システムの相違点の一つに，一旦決定した予定の再スケジューリングを対象としていることが挙げられる．

3. 制約充足問題

3.1 分散重み付き制約充足問題

制約が強過ぎて（過制約）解が存在しない場合に，問題を扱えるように CSP を拡張した枠組の 1 つが重み付き CSP である．重み付き CSP では，各制約に重みを与え，充足できない制約の重みの集合を最小にする解を探索する．この重み付き CSP をさらに分散環境へ拡張した枠組が，分散重み付き CSP [3] である．この CSP は，重み付き CSP が複数のエージェント上に分散された問題である．分散重み付き CSP は，$DVP = (P, P_I)$ によって定義する．$P = \{P_1, ..., P_m\}$ は重み付き CSP の集合である．P の各重み付き CSP $P_i = (X_i, D_i, C_i, S, \varphi_i)$ は，変数集合 X_i，変域集合 D_i，制約集合 C_i，評価構造 $S = (E, \otimes, \succ)$ および評価関数 $\varphi_i : C \rightarrow E$ を持つ．ここで，E は評価値の集合，$\succ$ は E 上で全順序である．$\otimes$ は評価値を総計するオペレーションである．$P_I = (X_I, D_I, C_I, S, \varphi_I)$ は複数のエージェントに関係する重み付き CSP で，変数集合 X_I $(\bigcup_{i=1}^{m} Xi$ の部分集合)，変域集合 D_I，制約集合 C_I，評価構造 S，および評価関数 φ_I を持つ．$\mathcal{A}$ は全変数への値割当てとする．分散重み付き CSP では，制約 c に関する $\mathcal{A}$ の評価値を $\varphi(\mathcal{A}, c)$ とすると，全変数の値割当ての評価値は，$\varphi(\mathcal{A}) = \otimes_{c \in C} \varphi(\mathcal{A}, c)$ と定義される．ここで，$C = (\bigcup_{i=1}^{m} Ci) \cup C_I$ は全制約集合である．

3.2 動的制約充足問題

我々は，カレンダーシステムにおける再スケジューリングを考えるために，動的 CSP の考えかたを導入している．動的 CSP [4] は，通常の CSP の列として表される．それぞれの CSP は，一つ前の CSP に対して制約の付加あるいは削除などにより変更を加えたものになる．

動的制約充足問題は以下のように定義される．

$$P = \{P_0, P_1, ..., P_i, ...\}$$

P は動的 CSP を表す．P_i が通常の CSP であり，それぞれの P_i は P_{i-1} に対して変化を加えたものとなる．

本稿では，制約にはそれぞれ重みが付加されているものとし，VCSP として定式化してきた．この場合にも，動的かつ重み付きの CSP としての定義は次のように簡単なものとして表現できる．

$$VP = \{VP_0, VP_1, ..., VP_i, ...\}$$

つまり，VP_i がそれぞれ 一つの VCSP となる．VCSP の場合には，各 CSP の変化として制約の重みの変更が考えられる．これは，重み変更前の制約の削除と新な重み付きの制約の追加と考えることもできる．

4. マルチエージェントによるカレンダースケジューリング

分散重み付き CSP では，エージェント間で制約を持つ変数以外の変数は各エージェント内で保持され，他のエージェントが正確に情報を得ることが出来ない．このため，個人のカレンダーを統合して利用する場合にプライベートな情報をより確実に保護することが出来る．このような背景から，参考文献 [1] では会議日程のスケジューリング問題に重み付き CSP を適用している．この会議日程スケジューリングでは，既に決定している日程が変更されることは好ましくないとし，既存の日程を単に制約として扱っている．しかし本稿では，個人のカレンダーに再スケジューリング可能なイベントを含まれるものとする．例えば，制約として現在の時刻から 3 日以内のイベントは変更不可，等の条件を指定する．また，参考文献 [1] で用いられている手法は，効率的に妥当な解をえることができるが，エージェント間での通信を行う際に一部プライベートなイベントに関する制約の情報を提供しなくてはならない．しかし本システムでは，プライベートな情報を完全に保護するために，greedy repair distributed method[3] を利用する．このアルゴリズムでは，エージェント間変数以外の変数は外部に情報が与えられることが無ため，プライベートな情報を各実に保護することができる．本システムにおけるエージェント間変数にあたる値は，複数ユーザ間で共有されるイベントである．そのため，個人のプライベートなカレンダーデータは完全に保護されることになる．

4.1 制約の種類

本システムでは既に存在していたイベントについても再度スケジューリングし直すことで，より適切なスケジューリング結果をえることを目的としている．しかし，このような再スケジューリングが何度も繰り返された場合，カレンダーの変動が激しいとユーザを混乱させてしまうことになる．そこで，ユーザが自身の意向を反映させるために追加する制約の他に，再スケジューリング後のデータはスケジューリング前と比較して似ているほど好ましいとする制約を始めからもっているものとする．このように本システムでは，ユーザが自身の意思で追加する制約の他に，より現実的なスケジューリング結果を得るためにシステムにあらかじめ搭載されている制約が存在する．このような制約の重用度は，基本的にユーザの設定した制約よりも重用度は低く設定されており，システムによる制約によりユーザの意思が妨げられることはない．

4.2 スケジューリングのプロセス

スケジュールを構成する要素であるイベントは e_{ij}(エージェント i のイベント j) と表す．e_{ij} は，開始日時 st，終了日時 et $(st \in T, et \in T, st < et)$，および内容情報 N_t の 3 つの要素で構成される．ここで，T は日時を表す離散値集合である．制約 c_{ij} はシステムに対応するフォーマットで記述し，イベント e_{ij} と制約 c_{ij} にはユーザによって 1 から 9 の重みが付けられる．提案されるイベント，及び再スケジューリングによる変更が可能なイベントは，日程の候補になる時間帯 $T_s = \{t_1, t_2, \cdots, t_n\}(t_i = [x, y](x, y \in T, x < y))$ と制約の重みの閾値 $T_r \in [1, 7]$，及びイベントに関する情報 N_t で構成される．

我々は，日程スケジューリング問題を分散重み付き CSP $DP = (P, P_I)$ として定義した．$P = \{P_1, ..., P_m\}$ は各個人エージェントの重み付き CSP の集合である．エージェント i の重み付き CSP $P_i = (X_i, D_i, C_i, S, \varphi_i)$ は，各イベント e_{ij} の開始日時 st と終了日時 et を変数集合 X_i とし，その領域集合 D_i は日時集合 T，制約集合 C_i はそれぞ

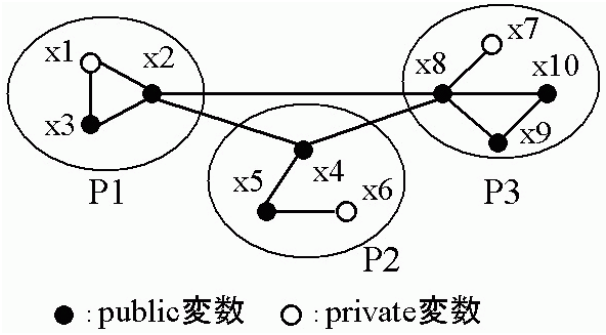


図 2: 分散重み付き CSP の例

れのイベントについての制約の集合である．評価構造 S は，$E = [0, 9], \succ =>, \otimes = +$ で，評価関数 φ_i は，違反制約の重みの和で表される．$PI = (X_I, D_I, C_I, S, \varphi_I)$ はエージェント間の重み付き CSP である．X_I は複数エージェント間で共有される変数であり，その領域集合 D_I は日時集合 T である．C_I はこれらのエージェント間に存在する制約の集合であり，評価関数 φ_I は基本的に φ_i と同様であるものとするが、CSP 全体の評価を求めるにあたって、単純に重みの和をとる評価関数 φ に加えて、以下の φ' を考慮する。

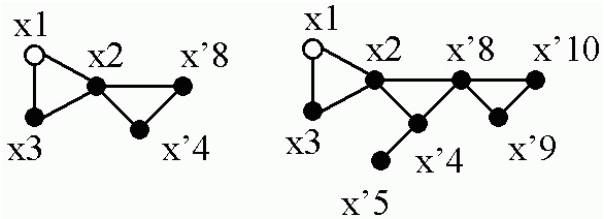
$$\varphi'(\mathcal{A}) = \max_i \varphi_i(\mathcal{A}),$$

つまり、最も評価値の悪いエージェントの評価値を考える。ここで、二つの評価値の優先度を $\varphi' > \varphi$ とする。全てのエージェントの評価値を小くし、その上で全体の和を小くする。

以上のように定式化した分散重み付き CSP の例を図 2 に示す．P_1, P_2, P_3 はそれぞれ 1 つのエージェントが持つ重み付き CSP である．図中の黒いノードはカレンダーにおいてパブリックな変数を表し，白いノードがプライベートな変数を示す．そして各ノードを結ぶ辺がそれぞれの変数間の制約である．このとき，複数のエージェントに渡る制約がエージェント間制約であり，その制約に関連する変数がエージェント間変数である．これらは必ず P_I にも含まれる．

我々は，DVCSP として定式化したミーティング日程のスケジューリングに対し，greedy repair distributed method[3] を用いる．ただし，この手法が P_I を構成する際に，X_I としてエージェント間変数しか考慮しないのに対し，我々は，システムのサーバーエージェントに公開されたイベントに関する変数全てが X_I に属するものとする．以下，本システムにおけるスケジューリングアルゴリズムの各ステップについて述べる．

- 各エージェントは，X_i に対する初期値の割り当て $\mathcal{A}_i$ を作成し，リーダーエージェントにその評価値を送信する．このためリーダーエージェントは全体での評価値を計算することが出来る．また，各 P_i 中に全 X_I のリモートコピー X'_I を作成する．
- リーダーエージェントはローカル変数とリモートコピーからなる重み付き CSP の割り当てを更新し，分散重み付き CSP 全体の評価値 $\mathcal{A}$ を再計算する．この評価値と更新された X_I の情報を各エージェントに送信し，次のエージェントにリーダーの権利を渡す．各エージェントはこの手続きを繰り返す．
- 評価値の更新がユーザの指定する範囲に収まった時点で終了する．



(a) X1' = エージェント間変数のみの場合　(b) X1' = public変数全体とした場合

図 3: 1 エージェントあたりの重み付き CSP

本システムでは，変数のリモートコピーを作成する際，greedy repair distributed method の場合に比べ，分散 CSP 全体に対し，各エージェントはより広範囲を把握することが出来る．同手法では，各 CSP の解決アルゴリズムとしては山登り法的な方法を用いている．そのため，公開されている変数のリモートコピーを追加することで，各エージェントの探索が局所的になることを改善している．

1 エージェントが持つ重み付き CSP の違いを，図 2 における $P1$ に相当するエージェントのリモートコピー作成後の重み付き CSP($P1'$) を例として図 3 に示す．図 3(a) が greedy repair distributed method の場合の $P1'$ の例であり，図 3(b) が本システムにおける $P1'$ の例である．

5. 動的 CSP による再スケジューリング

本システムでは，突然の予定に対するスケジューリングが行われる場合を考慮する．この時，既存のイベントを固定されたものとして扱うことも可能であるが，その結果が最適である可能性は低い．そのため既存イベントを含めた再スケジューリングを実現する．再スケジューリングの際には以下のような要求が発生する．

解の安定性： 再スケジューリング後の割当て結果は，そのスケジューリングが行われる前の割り当てに類似していることが好ましい．なぜならば，ユーザにとって改善の得られない予定変更は一般的に喜ばしいものではないだけでなく，混乱を招く恐れがある．

解の安定性の観点から，既存の動的 CSP の研究では次のような手法がとられている [5]．一つは，初期値利用法と呼ばれるもので，以前の問題の解を次なる問題における初期値として利用し，段階的にその値を改善して行く方法である．もうひとつは制約記録法と呼ばれるもので，以前の問題の解を求める過程で導かれた新な制約と，その制約が成立するための前提条件を記録しておく方法である．この手法には，制約が変化した場合に前提条件が無効となったものを取り除く必要がある．

以上の手法は通常の CSP を解くためにも広く用いられている手法である．しかし本稿では重み付き CSP として問題を定式化しているため，解の安定性を得るために，以前の解と同じ値をもつという重み付き制約を付加する方法をとる．本手法は，重み付き CSP が過制約な問題を扱うフレームワークであるからこそ実現できるものである．ただしこのとき，再スケジューリング時に新たに増える制約は，安定性を得るための制約より明かに重みが大きい必要がある． 本システムにおける動的重み付き CSP の解決の過程を以下に示す．

Step 1: カレンダーに新たに追加されたイベント (再スケジューリングの切っ掛けになるイベント) に関する制約を CSP に追加する．

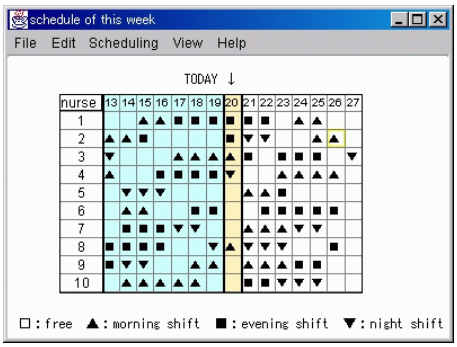


図 4: 看護勤務表の例

Step 2: 既存のイベント全てについて，前回の結果と同一であるという制約を，CSP に追加する．この制約は重みが再スケジューリング時の日時との近さによって決定され，近いほど大きく過去の値は変更できない．また，Step 1 で加えられた制約の重みを越えることはない．

Step 3: 上記の過程で制約が追加された CSP を解く

Step 4: 安定性を得るために追加された制約のうち，満たされた制約を CSP から削除する．

Step 4 の操作は，DVCSP の評価関数を 4.2 節の様に定義したことと関係がある．まず，Step 3 において問題を解決する際，予定の変更を余儀無くされたエージェントは，Step 2 での制約の分だけ評価値が悪くなる．ここで，4.2 節において DVCSP の評価値は最も大きい評価値をとる CSP の値としたため，DVCSP 全体での最適解を求めるために、割り当て値が変わったエージェントの他の制約は満される傾向が高くなる．Step 4 において制約を削除するのは，スケジューリング時の日付けにより重みを毎回指定し直すためであり，違反された制約を残すのは，予定を変更されたエージェントに対して次回以降のスケジューリングにおいても他の制約について優遇するためである．

6. ナーススケジューリングへの適用

本システムの具体的な応用の一つとして，ナーススケジューリングへの適用が考えられる．既存のナーススケジューリングに関する研究では，勤務表の再スケジューリングが考えられていない．それは現実に病院でそのように勤務が割り当てられているためである．しかし，看護師に急な予定が入る場合も十分考えられるため，本システムの応用分野として考える．

6.1 ナーススケジューリング問題

本稿では，看護勤務表として図 4 のようなものを考える．勤務表中の一行は看護師のスケジュールを表し，列は日付を表す．[2] では，この問題に対して次の様に定式化している．CSP(X, D, C) のうち，変数 X として，表中のセルを割当る．つまり，一人の看護婦の 1 日の予定が変数となる．D には休日，朝，夕方，深夜の 3 つの勤務形態と休日を考えている．この問題を本システムに適用するには，3.1 節における日付集合 T に対して日付の値のみを考え，4.2 節のイベント e_t については内容情報 N_t として勤務形態の情報を与えることにより実現する．

6.2 DVCSP としてのナーススケジューリング

ナーススケジューリングを DVCSP とする場合，各ナースの勤務毎に CSP を考えることができる．この時，以下の制約がエージェント内の制約として考えられる．一つは対応する看護師の各勤務の回数に関する制約であり，禁止されている勤務の並びなども考慮される．また，休日などの要望に関する制約もエージェント内の制約として考えられる．

変数はほとんどの変数がエージェント間で参照されることになる．エージェント間制約として，一日の各勤務の人数に関する制約及び勤務に就く看護師のベテラン，新人の割合に関する制約などが考えらえる．ナーススケジューリング問題については，DVCSP のプライバシーに関する特長はあまり意味をなさない．しかし，各 CSP の評価関数とエージェント間 CSP の評価関数を分けて考えることが可能となり，本システムのように，勤務予定の変更を依頼された看護師の他の要望を優遇することが可能となる．

7. おわりに

分散重み付き CSP に基づくカレンダースケジューリングを提案し，マルチエージェントカレンダーシステムを実現した．カレンダーに対するスケジューリングを実現するにあたって，動的 CSP による再スケジューリングを考え，既定のイベントをも対象とするスケジューリングを実現した．また，分散 CSP に基いているため，ユーザのプライベート情報を公開せずに，個人の要望を反映することが可能となった．

現時点での課題として，DVCSP の評価を最も評価値の悪いエージェントの値のみを考慮し，それを満せばあとは全制約の重みの和で判断していることが挙げられる．このため，一つだけどうしても評価値が低くなるエージェントがいる場合には，実質的に重みの和のみで判断することになってしまう．これを解決するためには，エージェント間 CSP の評価関数を改善する必要がある．

参考文献

[1] Takuo Tsuruta, and Toramatsu Shintani “Scheduling Meetings Using Distributed Valued Constraint Satisfaction Algorithm,”In the Proceedings of the Fourteenth European Conference on Artificial inteligence(ECAI2000),pp.283-387,Aug.2000.

[2] S. Abdennadher and H. Schlenker “Nurse Scheduling using Constraint Logic Programming.” Eleventh Annual Conference on Innovative Applications of Artificial Intelligence, IAAI-99, (1999)

[3] M.Lemaitre, and G.Verfaillie, “An incomplete method for solving distributed valued constraint satisfaction, ” In Proc. of AAAI-97 Workshop on Constraints and Agents,(1997)

[4] Dechter, R., and Dechter, A. “Belief Maintenance in Dynamic Constraint Networks.” In Proc. of AAAI-88 (1988), 37–42

[5] 横尾真，平山勝敏，“CSP の新しい展開：分散／動的／不完全 CSP ，” 人工知能学会誌，Vol. 12，No. 3，pp. 381-389，1997.