Automatic Problem Generation in Constraint-Based Tutors

Brent Martin, Antonija Mitrovic: Automatic Problem Generation in Constraint-Based Tutors, Proceedings of the Sixth International Conference on Intelligent Tutoring Systems, pp.388-398 (2002).

制約に基づくモデル化 (CBM: Constraint-Based Modeling) は，簡単に成熟する生徒のモデル化技術である．我々は CBM を用いたチュータをいくつか実装し，特に open-end な領域において適切であることを示してきた．open-end で複雑な領域のモデルの問題は，拡張された学習セッションで充分な練習を供給するために包括的な問題セットを必要とする，その大きいサイズである．我々は，領域知識から直接的に新しい問題を自動的に生成するアルゴリズムを開発することで，この問題に取り組んだ．我々はこのアルゴリズムを示し，生成された問題を使う生徒と教師が制作した問題セットを使う生徒のパフォーマンスを比較し，生成された問題セットのパフォーマンスが優れていることを示す．

1 Introduction

□ SQL チュータ [Mitrovic 1998]

- CBM を使用した SQL データベース言語教育 ITS
  - CBM: 領域・生徒モデルの構築を容易に
- 教示セッションを（生徒にあわせて）仕立て
  - 生徒の解入力時にフィードバック
  - 問題の難易度のコントロール
  - scaffolding 情報を供与
- 問題生成機能の実装
  - 問題切れへの対応
  - 領域モデルから生徒モデルに適合する新しい問題を生成

□本稿の構成

- SQL チュータの実装
- CBM と問題生成機能
- 新しい問題セットの 6 週間の分析
- まとめと今後の展望

2 SQL-Tutor

- Canterbury 大学二年・三年の SQL データベース検索言語を教育
- CBM
  - 領域を状態の制約でモデル化
    If ＜関連条件＞が生徒の解法において真
    $THEN$ ＜充足条件＞も真
  - 生徒の教育上の状態を関連条件でテスト
    関連条件が満たされる → 充足条件をチェック
    満たされない → 生徒は誤答・フィードバックを与える
  - 利点
    :モデルが不完全でもよい，open-end な領域にも使用可能
    :モデルトレース [Anderson 1995] 等に対するアドバンテージ
- 制約表現
  - Lisp fragment によるパターンマッチングで実装

```
(147
 ''WHERE 節にこのデータベースにない名前を使用している''
    (match SS WHERE (?* (^name ?n) ?*))
    (or (test SS (^valid-table (?n ?t)))
        (test SS (^attribute-p (?n ?a ?t))))
 ''WHERE'')
```

  - 関数 MATCH: (match ＜解法名＞ ＜節＞ ＜パターンのリスト＞)

解法名：「SS（生徒の解法）」か「IS（模範解）」
節：SQL の節（「**SELECT**」等）
パターンのリスト：テストする解法要素
マクロ（`^valid-table` 等）：別定義の関数を呼び出し
- 関数 TEST: MATCH の特殊系・変数の値を評価
- 生徒の入力からの解法生成
:充足された制約から妥当なマッチの fragment
:破られた制約から妥当でない fragment
→ 模範解のマッチする fragment から修正

- 修繕理論 (Repair Theory) との違い [VanLehn 1983]
  - 人間の振舞いのモデル化ではない
  - 本手法の利点：エラー検出が不要・計算爆発しない

3 Generating New Problems

□問題選択

- 難易度・生徒が苦労している概念に基づく選択
  - よく破られる制約の集合を検索
  - 検索された制約と関連する問題セットを取り出し
  - 生徒のレベルに合った問題を選択

- 問題供給の問題
  - 制約数（500 以上）に応じた多量の問題が必要
  - 6 週間の実用評価における解かれた問題数・適用された制約（→ Fig.1）
:40 問までは問題数に対し制約数が線形増加 ($R^2 = .83$)
:問題を沢山解くと新しい制約の出現がなくなる
→ 各制約毎に問題を多数用意する必要あり
（問題のバリエーションが 40 問で頭打ちになった？）

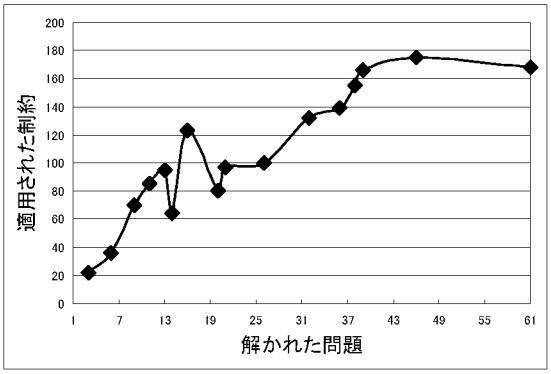

Fig.1 新しい適用された制約

- 問題生成
  - 制約集合からの新しい問題（模範解）を生成
  - ユーザが解を自然言語の問題文に変換

□新しい模範解の構築

- 特定の制約からの新しい模範解の生成
  - 制約の fragment を模範解（この段階では空）に挿入
  - 変数の具体化
関係操作：$>, <, <=, >=, =, <>, != $からランダム選択
その他（文字列）：具体化制約

```
(I8
 ''WHERE 節の文字列比較に妥当な文字列があることを確認する''
    (match SS WHERE (?* (^attribute-of (?n ?a ?t) ''='' (^sql-stringp ?s)  ?*))
    (test SS (^valid-string (?s ?a ?t))))
 ''WHERE'')
```

`^valid-string`：属性名と文字列を列挙したマクロ
- 生成された SQL 文の syntax チェック・修正

□新しい問題セットの作成

- 問題生成の方法
  - 生徒モデルにあわせてその場で生成 → 問題文生成ができません
  - 問題セットを事前に作成 → 本研究はこちら
- 問題生成の実行
  - 800 の模範解を与え，制約 1 つごとに 1 問を生成
  - 最良のものを選択・問題文に変換
  - 3 時間頑張って 200 問を生成
    :人手だと何日もかかって 82 問
    :Fig.1 の cut-off が 40 問から 60 問に → 新しい問題セットが増殖された

4 Evaluation

□評価方法

- 問題生成の目的
  - 時間のかかる作業（問題セットを書く）の自動化によるシステム構築の手間軽減
- この目的において達成すべき項目
  1. アルゴリズムが動作する（新しい問題を生成できる）
  2. 人の携わる部分が伝統的な問題作成より少ない
  3. 生成された問題が人手で作成された問題と同様程度に学習を促進する

  → 1・2 は確認済み
- SQL チュータの評価実験
  - 被験者：大学 2 年生のデータベース受講生
  - 被験者群
    統制：人手で作成した問題を積んだシステムで学習
    実験：システムが生成した問題を積んだシステムで学習
    ?: 生徒モデル視覚化のシステム（他の研究）で学習
  - システムの使用
    :講義の評価の一環として，6 週の期間中好きなだけ使用可能
    :使用時間：平均 2 時間半（0 〜何時間も）
    :解いた問題数：平均 25 問（0 〜 98）
    （システムを使用しなかった被験者を除外，統制 24 名・実験 26 名）
  - システム使用前に事前テスト・最後に事後テストを実施
- 評価方法
  - 事前テスト・事後テストの成績
    :事前テストの比較により群間に差がないことを確認
  - 被験者の学習時のエラー数
    :間違えた制約（＝学習領域における概念）
    :エラー数グラフの勾配で 2 群間の学習速度を比較
  - 被験者が見つけた問題の難しさ
    :システムが生成した問題の難易度の適切さ
    :被験者が諦めた問題数の群間比較により評価
    （問題が簡単過ぎる・難し過ぎる場合，違いタイプの問題が欲しい場合）

□問題の難易度

- 問題を解くことを諦めた時の理由
  - 被験者のシステム利用の記録
- 結果（→ Table 1）
  - 諦めた：被験者が解くことを諦めた問題の割合
    諦めた理由：難し過ぎる・簡単過ぎる・違うタイプの問題が欲しい
  - 回答あり：諦めた問題の中の理由を答えてくれた問題の割合

↓

- 統制群・実験群の間に差なし
→ システムが生成した問題は難易度に問題なし

Table 1. 諦めた問題

群	諦めた (%)	難し過 (%)	簡単過 (%)	違タイプ (%)	回答あり (%)
統制	26	24	42	34	84
実験	26	22	42	35	62

- 各問題を解く際の試行数（→ Table 2）
  - 解決/生徒：正しく解かれた問題数の平均
  - 合計時間：システムを利用した合計時間の平均
  - 試行/問題：入力された試行数の平均
  - 時間/問題：1 問に費やされた平均時間
  ↓
  - 統制群・実験群の間に差なし
  → どちらのシステムも被験者に同様に扱われた

Table 2. 問題毎の試行と時間

群	解決/生徒	合計時間 (h)	試行/問題	時間/問題 (m)	事前テスト
統制	23	2:37	3.96(1.9)	6:14(3.6)	4.82(1.5)
実験	26	2:31	3.45(1.2)	5:50(3.2)	5.06(1.5)
優位？			NO	NO	NO
T 検定			p=.31	p=.71	p=.48

□学習速度

- 算出方法
  - n 番目の問題の解決で破られた制約の割合をプロット
- べき曲線の適合（→ Figure 2）
  - cut-off は 5 問
  - n=1 での傾きは実験群が優位に大きい （→ Table 3）

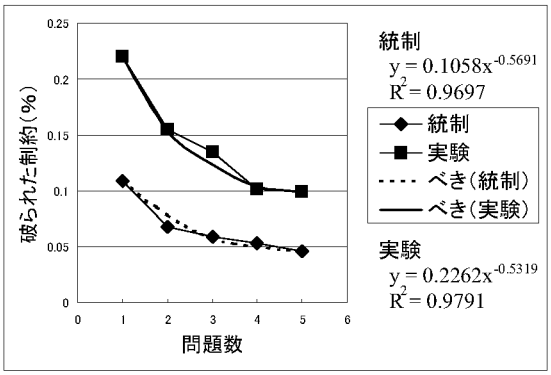

Fig.2 学習パフォーマンス

Table 3. 初期の学習速度

群	傾き	適合 (R^2)
統制	.07(.04)	.63(.29)
実験	.16(.12)	.68(.30)
T 検定	YES(p=.01)	NO(p=.61)

5 Conclusions

- 練習問題の供給の問題

- 制約に基づくチュータにおける解決
  - 領域モデルからの問題生成
- SQL チュータの評価実験
  - 人手で作成した問題と同程度の効果
- 元気が出る結果
  - システムが生成した問題は学習速度を向上
  - 問題の数？質？

6 Further Work

- 開発 CBM の統合
- 問題文変換の設計

資料：SQL について

SQL（Structured English QUery Language: SEQUEL がさらに略された語）とは，関係（要するに表）操作の形式的な表現で，関係代数だの関係論理だのといったもののややこしい操作を人にとって平易にしたユーザインタフェース．

2005 年セリーグ成績表

順位	チーム	試合数	勝数	負数	引分	勝率	ゲーム差	得点	失点	本塁打	打率	防御率
1	阪神	146	87	54	5	.617	-	731	533	140	.274	3.24
2	中日	146	79	66	1	.545	10	680	628	139	.269	4.13
3	横浜	146	69	70	7	.496	7	621	596	143	.265	3.68
4	ヤクルト	146	71	73	2	.493	0.5	591	596	128	.276	4.00
5	巨人	146	62	80	4	.437	8	617	737	186	.260	4.80
6	広島	146	58	84	4	.408	4	615	779	184	.275	4.80

例 1：2005 年のセリーグで，チーム本塁打が 150 本以上のチームの順位は？
「2005 年セリーグ成績表」から「本塁打が 150 以上である」行の「順位」を取り出す

↓

Select 順位
From 2005 年セリーグ成績表
where 本塁打 $\geq$150

↓

順位
5
6