

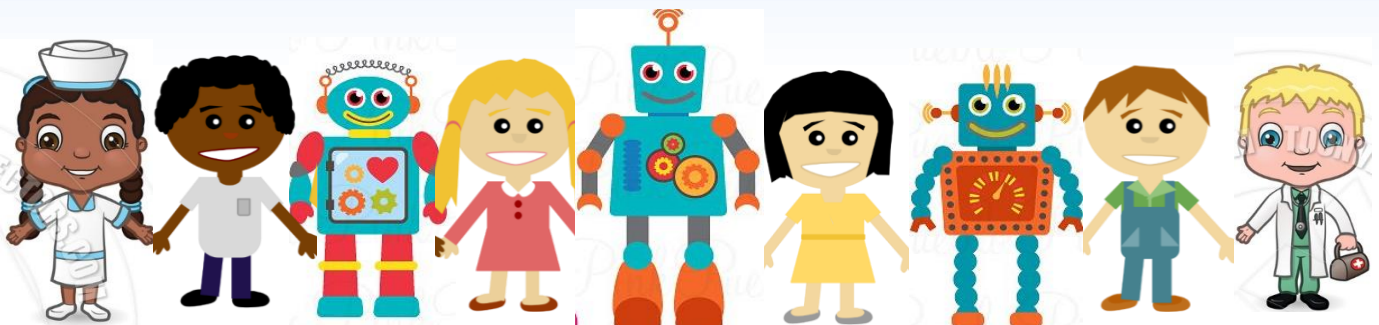


Pepper Project Group

CAO LIANG	A0012884E
GENG LIANGYU	A0195278M
HAN DONGCHOU FRANCIS	A0195414A
ONG BOON PING	A0195172B
TAN CHIN GEE	A0195296M

Patient Matching System Depression Screening System

Project Report



Contents

- Executive Summary	3
- Business Problem Background	4
- Project Objectives & Success Measurement	5
- Project Solution	6
- Project Implementation	10
- Project Performance & Validation	28
- Project Conclusions: Findings & Recommendation	30

Executive Summary

This project is a continuation from the Depression Screening System group project completed during the Machine Reasoning course. As a recap, the Depression Screening System deals with the problem of depression in youths and vulnerable segments of the community going undetected and untreated. It serves as an early warning system to put a spotlight on those vulnerable individuals who are displaying symptoms of depression. These vulnerable individuals can be spotted through the PHQ-9 Survey framework devised with the Depression Screening System. The output from that system is to provide a diagnosis (with a PHQ-9 score) on the level of severity of depression the individuals are having.

In this phase of project, the objective is to use the PHQ-9 scores of the individuals and match them to the many practitioners in the community partners, hospitals and IMH.

Why Patient Matching? When we churn out a list of patients in batches as part of the PHQ-9 Survey, there is also a need to do the matching with practitioners in batches. There are some challenges to match patients to practitioners. First of all, practitioners fall into roughly 3 categories – Counsellors, Psychologists, Psychiatrists. Then there is the time availability of the practitioners, as well as language ability, which are all hard constraints. Also there is location preference and gender preference which are soft constraints. Then there is the cost based on the practitioner selected, which is a soft constraint to minimize it.

Using the State Space Search techniques embedded in the OptaPlanner, the Patient Matching System is able to optimally match a group of patients against a group of practitioners. The detail of how this match is executed will be covered in the next chapters.

This system provides benefits to those who are diagnosed with depression by matching appropriate practitioners with the right expertise and in accordance to some set constraints.

Business Problem Background

In the Depression Screening System, the test candidates are given the PHQ-9 scores indicating the level of severity of the depression.

With the current project, the Patient Matching System matches patients with mental health practitioners based on two hard constraints and four soft constraints. The hard constraints are patients' depression severity and language ability. The soft constraints are patients' preferences for the day of the appointment, where the practitioners are located, gender of the practitioners, and treatment costs.

Starting with the hard constraints, our system scores the severity of a patient's depression with a number from 0 to 27. Patients with scores between 10 to 27 require treatment. A score between 10 to 14 indicates that a patient is moderately depressed, and the patient should see a counsellor. A score between 15 to 19 indicates that a patient's depression is moderately severe, and the patient should see a psychologist. A score between 20 to 27 indicates that a patient is severely depressed, and the patient should see a psychiatrist.

Patients and mental health practitioners need to communicate and understand each other during treatment. The system pairs patients with mental health practitioners who share at least one language in common. Patients and mental health practitioners can speak one or more of these languages: English, Mandarin, Malay or Tamil.

Our system optimizes the matching of patients with practitioners through fulfilling four soft constraints. Patients will indicate a day of the week, Monday to Sunday, that they preferred to see the practitioners. The system will try to match patients with practitioners who are available on the preferred day.

Patients will indicate where they would prefer to see the practitioners. There are five regions to choose from: Central, East, North, Northeast, and West. The system will try to match patients with practitioners working in their preferred regions.

Patients will indicate whether they prefer to see female or male practitioners which the system will try to match accordingly.

Finally, the system will try to pair patients with practitioners that have lower treatment costs to keep the overall healthcare expenditure low.

Project Objectives & Success Measurement

The objective of this project is to optimally match patients with practitioners, both in groups. The outcome of this match will be such that all patients will be assigned matching practitioners.

We will be monitoring whether the system is able to meet the hard constraints and soft constraints during this matching exercise to see how good the application is in tackling hard constraints and minimizing soft ones. This will provide us with a gauge as to how well the application is able to perform the matches.

The hard constraints are

- 1) patients' depression severity
- 2) language ability

The soft constraints are

- 1) patients' preferences for the day of the appointment
- 2) where the practitioners are located
- 3) gender of the practitioners
- 4) treatment costs.

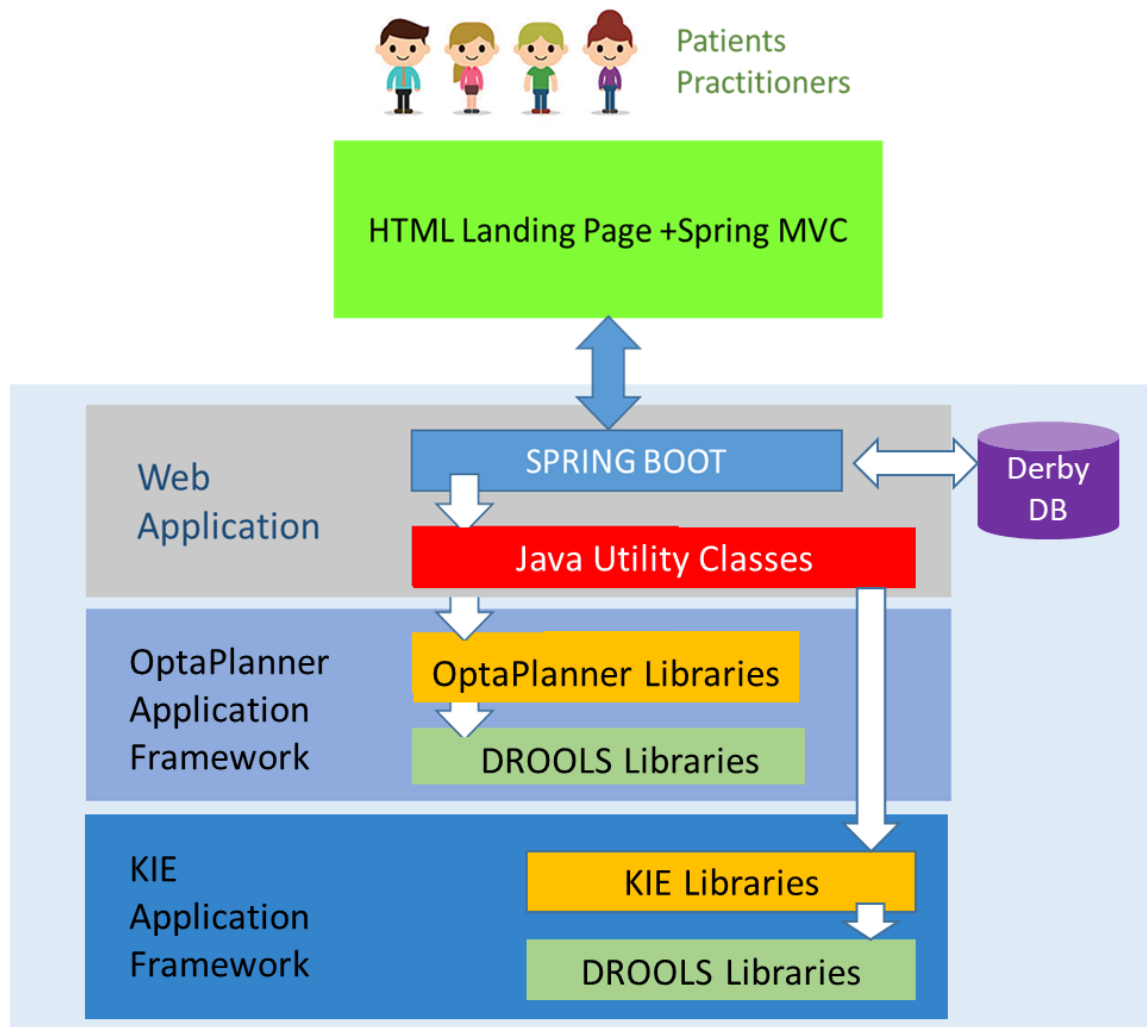
Project Solution

Architecture

This application leverages the combination of KIE Application Framework with Optaplanner libraries, DROOLS libraries, custom Java Classes and Spring Boot for development, testing and deployment. We utilized the KIE for metadata and workflow, DROOLS for rules engine, Optaplanner for matching the patients with practitioners and the JBOS Web Server to provide the web application server functions. In addition, we used the Spring Boot to program the web application, landing page, and interface with KIE.

A lot of effort was spent on integrating the Spring Boot web application with the KIE framework, DROOLS libraries and Optaplanner libraries through our self-coded Java classes.

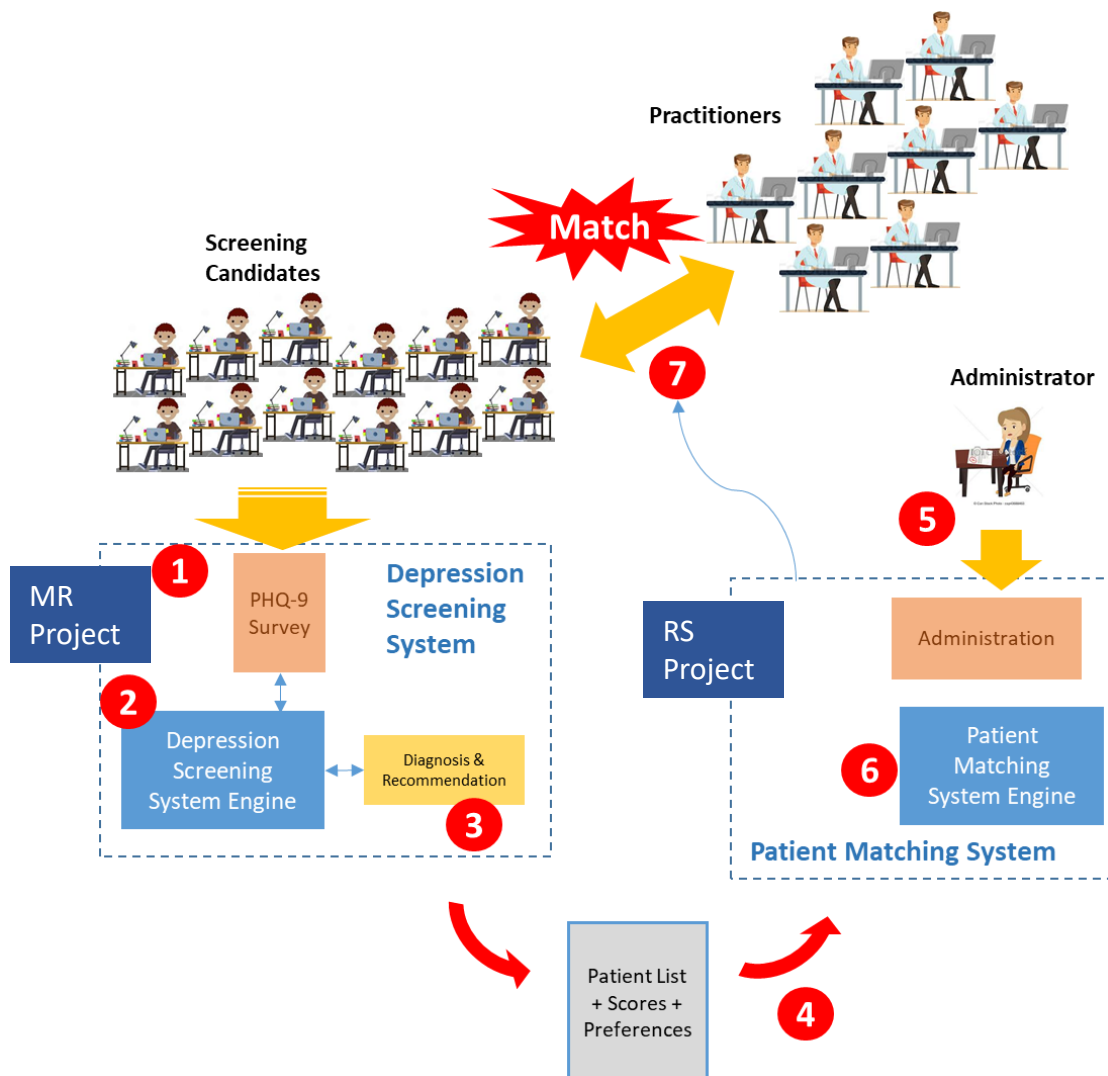
This allows the user to not only interface with an external HTML page but also stay on the HTML frontend throughout the entire process.



Project Solution

Process Flow

- 1) Candidates undergo screening process by taking part in the PHQ-9 online survey
- 2) The Rules Engine of the KIE Workbench churns out the scores for the candidates
- 3) Based on the scores, the candidates are grouped into the different severity levels
- 4) The list of patients and the respective scores and preferences are sent in batches to the Patient Matching System
- 5) An administrator can then activate a match
- 6) The Patient Matching System engine will start to leverage the OptaPlanner system
- 7) The patients are matched to the practitioners based on the hard and soft constraints



Project Solution

Design Strategy

In trying to establish the most optimal search method, we took reference from the following guideline found in the Optaplanner documentation (extract below).

Following this guideline, we adopted the First Fit Decreasing Search Method at the start and then tuned further for search improvements. Details are laid out in the subsequent slides.

Reference: <https://docs.optaplanner.org/6.1.0.Beta4/optaplanner-docs/html/optimizationAlgorithms.html>

Which optimization algorithms should I use?

The *best* optimization algorithms configuration for your use case depends heavily on your use case. Nevertheless, this vanilla recipe will get you into the game with a pretty good configuration, probably much better than what you're used to.

Start with a quick configuration that involves little or no configuration and optimization code:

First Fit

Next, implement planning entity difficulty comparison and turn it into:

First Fit Decreasing

Next, add Late Acceptance behind it:

1. First Fit Decreasing
2. Late Acceptance. A late acceptance size of 400 usually works well.

At this point *the free lunch is over*. The return on invested time lowers. The result is probably already more than good enough.

But you can do even better, at a lower return on invested time. Use the Benchmark and try a couple of different Tabu Search, Simulated Annealing and Late Acceptance configurations, for example:

- First Fit Decreasing
- Tabu Search. An entity tabu size of 7 usually works well.
- Use the Benchmark to improve the values for those size parameters.
- If it's worth your time, continue experimenting further. For example, you can even combine multiple algorithms together:
 - First Fit Decreasing
 - Late Acceptance (relatively long time)
 - Tabu Search (relatively short time)

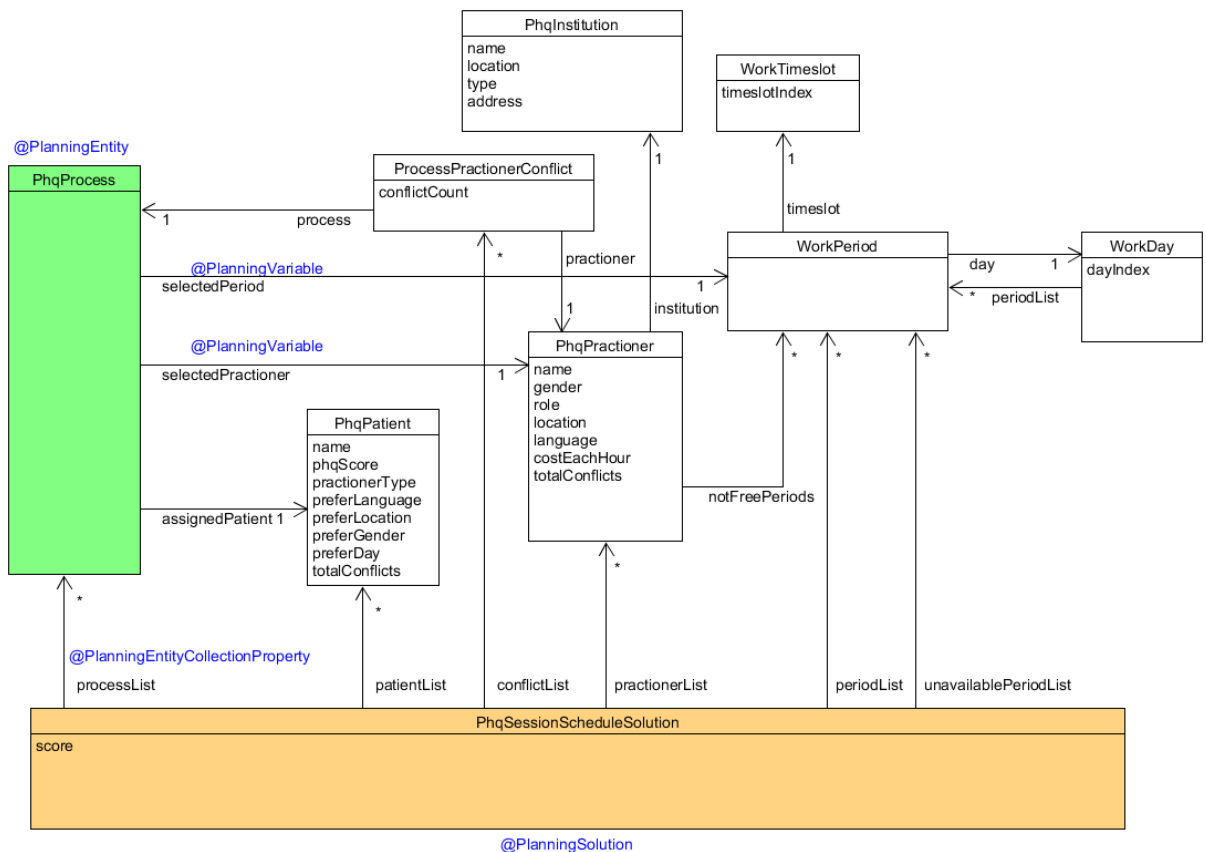
Project Solution

Design – Class Diagram

The following figure shows the Class Diagram for the application. This class diagram shows that, using OptaPlanner, we are trying to match a group of patients to a group of practitioners based on Period and Practitioner.

Following is the journey that we adopted.

- 1) We started on the KIE Workbench to prototype the application and then migrated to Eclipse and incorporated Java Maven libraries
 - A note that on the KIE environment, we realized that the memory process is intensive and the OptaPlanner is limited to few functions (eg cannot tune the late acceptance properties)
 - Also, using Java Maven libraries brought about some enhancement to the model
- 2) We evolved the Data Objects as we moved along, resulting in the following Class Diagram



Project Solution

Scoring Logic

HARD SCORE

One Patient One Slot

// One individual patient can only take one unique timeslot.

// If this is not met, then reduce the hard constraint score.

```
rule "practitionerSlot"
```

```
when
```

```
PhqProcess($id :id, $selectedPractitioner : selectedPractitioner, $selectedPeriod: selectedPeriod, selectedPractitioner != null,  
!(selectedPractitioner.nilPractitioner) )
```

```
PhqProcess(id <$id, $selectedPractitioner == selectedPractitioner, $selectedPeriod == selectedPeriod )
```

```
then
```

```
scoreHolder.addHardConstraintMatch(kcontext, -28);
```

```
end
```

Practitioner Type – Counsellor, Psychologist, Psychiatrist

// The selected practitioner must match the required practitioner type for the specific patient according to the patient's PHQ9 score.

// The adequate practitioner type for the patient is computed in patient.getPractitionerType() function.

// If this is not met, then reduce the hard constraint score.

```
rule "practitionerSelection"
```

```
when
```

```
PhqProcess($selectedPractitioner : selectedPractitioner, selectedPractitioner != null,
```

```
!(selectedPractitioner.nilPractitioner),
```

```
assignedPatient.practitionerType != selectedPractitioner.role)
```

```
then
```

```
scoreHolder.addHardConstraintMatch(kcontext, -28);
```

```
end
```

Project Solution

Scoring Logic

Language

// The selected practitioner must speak the language that matches the specific patient's preferred language.

// Since all the practitioners will be able to speak "English", the system will not check on "English",

// The patient must specify a preferred language or "English" must be selected.

// If this is not met, then reduce the hard constraint score.

rule "languageRule"

when

PhqProcess(\$selectedPractitioner : selectedPractitioner, selectedPractitioner != null,

!(selectedPractitioner.nilPractitioner),

assignedPatient.preferLanguage != 0, // Patient prefer language != "English"

assignedPatient.preferLanguage != selectedPractitioner.language)

then

scoreHolder.addHardConstraintMatch(kcontext, -28);

end

Practitioner's Availability

// The practitioner must be available at the selected period.

// If this is not met, then reduce the hard constraint score.

rule "practitionerUnavailableRule"

when

PhqProcess(\$selectedPractitioner : selectedPractitioner, selectedPractitioner != null,

!(selectedPractitioner.nilPractitioner), \$selectedPeriod : selectedPeriod, selectedPeriod != null,

selectedPractitioner.isUnavailablePeriod(selectedPeriod))

then

scoreHolder.addHardConstraintMatch(kcontext, -28);

end

Project Solution

Scoring Logic

Practitioner's Assignment

// When the system is able to allocate a practitioner, then award the hard constraint with positive increment.

// This is to push the system to assign an available practitioner, and not choose nilPractitioner which represents no available practitioner.

// At the same time, this also allows the system not to choose practitioner (i.e. choose nilPractitioner) in case other hard constraints are not met

// This rule also allows us to learn whether all the hard constraints are met by comparing the final hard constraint with the total PHQ9 points.

rule "phqRule"

when

PhqProcess(selectedPractitioner != null, !(selectedPractitioner.nilPractitioner), \$phqScore : assignedPatient.phqScore, \$phqScore>9)

then

scoreHolder.addHardConstraintMatch(kcontext, \$phqScore);

End

Project Solution

Scoring Logic

SOFT SCORE

Gender

//Not meeting the gender constraint does not generally lead to a serious impact to the therapy outcome. Hence, it is categorized as soft constraint in order to attempt a match. Weightage is set to 80 because the least cost of a session is \$80. This makes the soft score to be comparable to the cost.

```
rule "genderRule"
```

```
when
    PhqProcess($selectedPractitioner : selectedPractitioner, selectedPractitioner != null,
        assignedPatient.preferGender != 0, // Patient prefer gender != "Any"
        assignedPatient.preferGender == selectedPractitioner.gender)
then
    scoreHolder.addSoftConstraintMatch(kcontext, 80);
end
```

Preferred Location

//Patient's preferred location is a matter of convenience and does not adversely affect the outcome of the therapy. Hence, it is categorized as soft constraint to get the search algorithm to try. The weightage is same as gender.

```
rule "locationRule"
```

```
when
    PhqProcess($selectedPractitioner : selectedPractitioner, selectedPractitioner != null,
        assignedPatient.preferLocation != 0, // Patient prefer location != "Any"
        assignedPatient.preferLocation == selectedPractitioner.location)
then
    scoreHolder.addSoftConstraintMatch(kcontext, 80);
end
```

Project Solution

Scoring Logic

Preferred Day

//The patient may choose a preferred day for the session. It is not mandatory to match with the practitioners' slot, but a search attempt should be made. Hence, it is categorized as soft constraint. The weightage is same as gender.

```
rule "preferDayRule"

when
    PhqProcess($selectedPractioner : selectedPractioner, selectedPractioner != null,
        $selectedPeriod : selectedPeriod, selectedPeriod != null,
        assignedPatient.preferDay != null, // Patient prefer location != "Any"
        assignedPatient.preferDay == selectedPeriod.day)
then
    scoreHolder.addSoftConstraintMatch(kcontext, 80 );
end
```

Minimize Cost

//In the general healthcare landscape, the patient's cost should be minimized as much as possible. The highest fee is \$120, hence the soft score is the saving measured from highest cost.

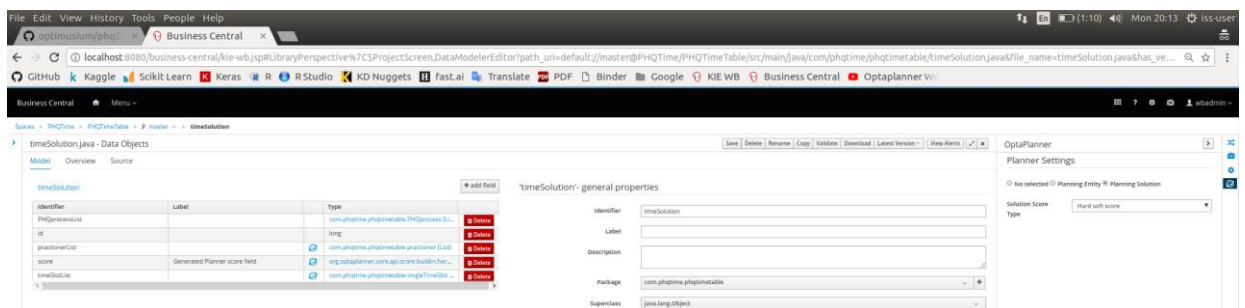
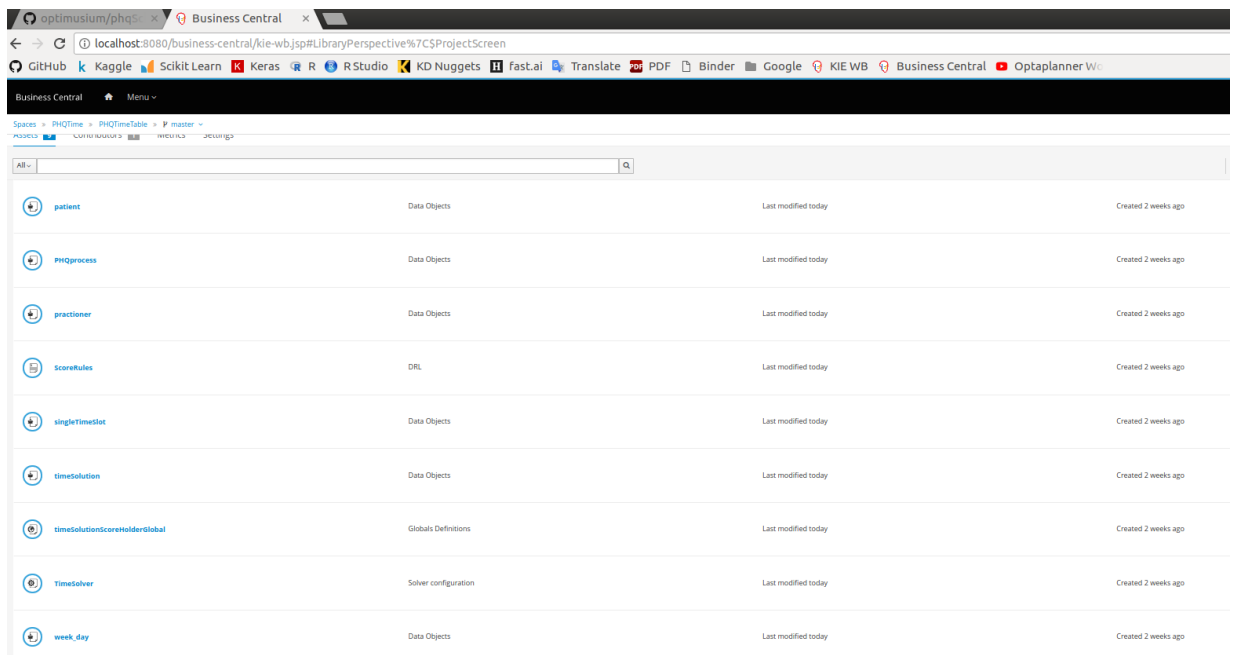
```
rule "costRule"

when
    PhqProcess($selectedPractioner : selectedPractioner, selectedPractioner != null)
then
    scoreHolder.addSoftConstraintMatch(kcontext, 120-$selectedPractioner.getCostEachHour());
end
```

Project Implementation

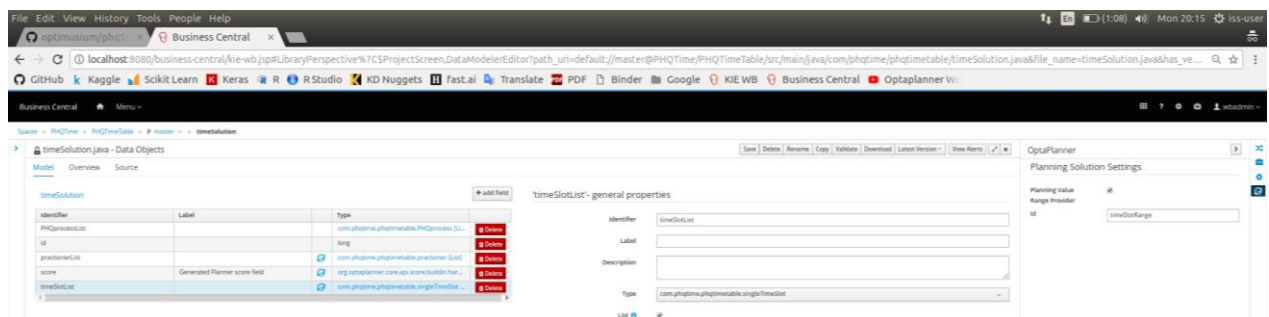
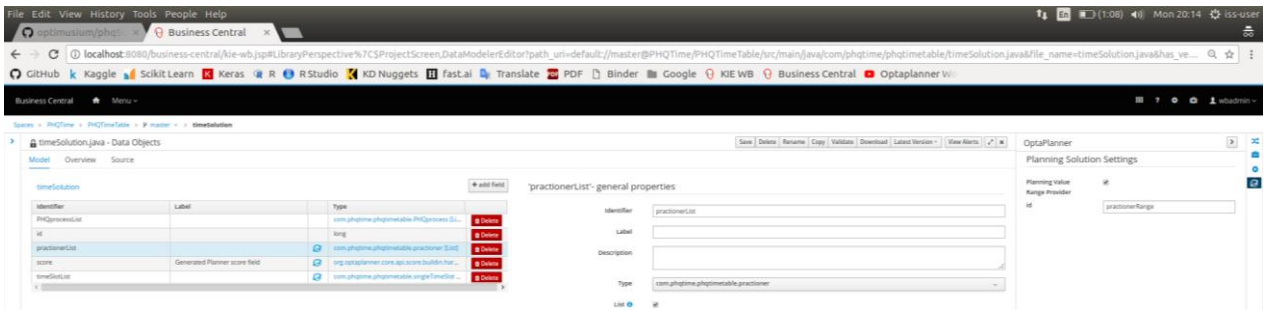
KIE Workbench Setup

The following screens shows the setup... - to update.



Project Implementation

KIE Workbench Setup



Project Implementation

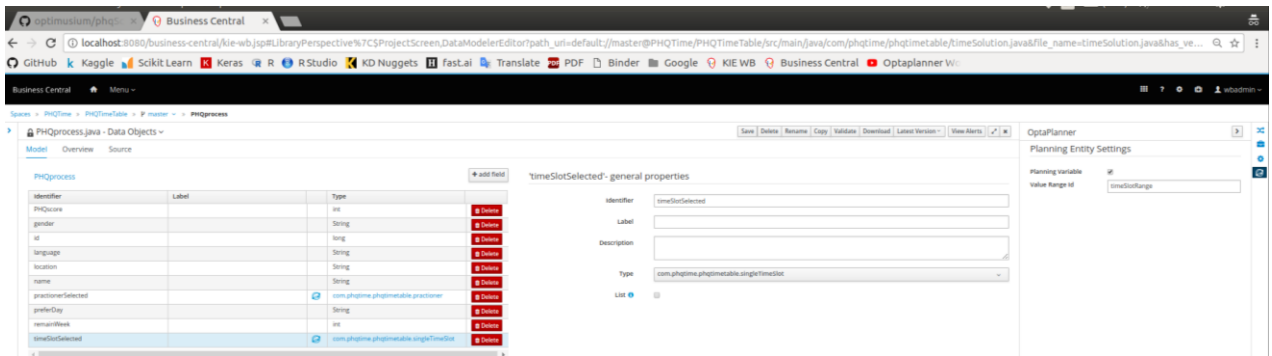
KIE Workbench Setup

The screenshot shows the KIE Workbench interface for editing the `PHQProcess` model. The left pane displays the `PHQProcess` data objects table, which includes fields like `id`, `gender`, `language`, `location`, `name`, `practitionerSelected`, `practitionerDay`, `remainingWeeks`, and `timeSlotSelected`. The right pane shows the `PHQProcess` general properties, including `Identifier`, `Label`, `Description`, `Package`, and `SuperClasses`. The `OptaPlanner` settings pane on the far right shows the `Planner Settings` for `PHQProcess`, including a `Planning Solution` and a `Difficulty Comparator`.

The screenshot shows the KIE Workbench interface for editing the `practitionerSelected` model. The left pane displays the `practitionerSelected` data objects table, which includes fields like `id`, `gender`, `language`, `location`, `name`, `practitionerSelected`, `practitionerDay`, `remainingWeeks`, and `timeSlotSelected`. The right pane shows the `practitionerSelected` general properties, including `Identifier`, `Label`, `Description`, `Type`, and `Link`. The `OptaPlanner` settings pane on the far right shows the `Planning Entity Settings` for `practitionerSelected`, including a `Planning Variable` and a `Value Range`.

Project Implementation

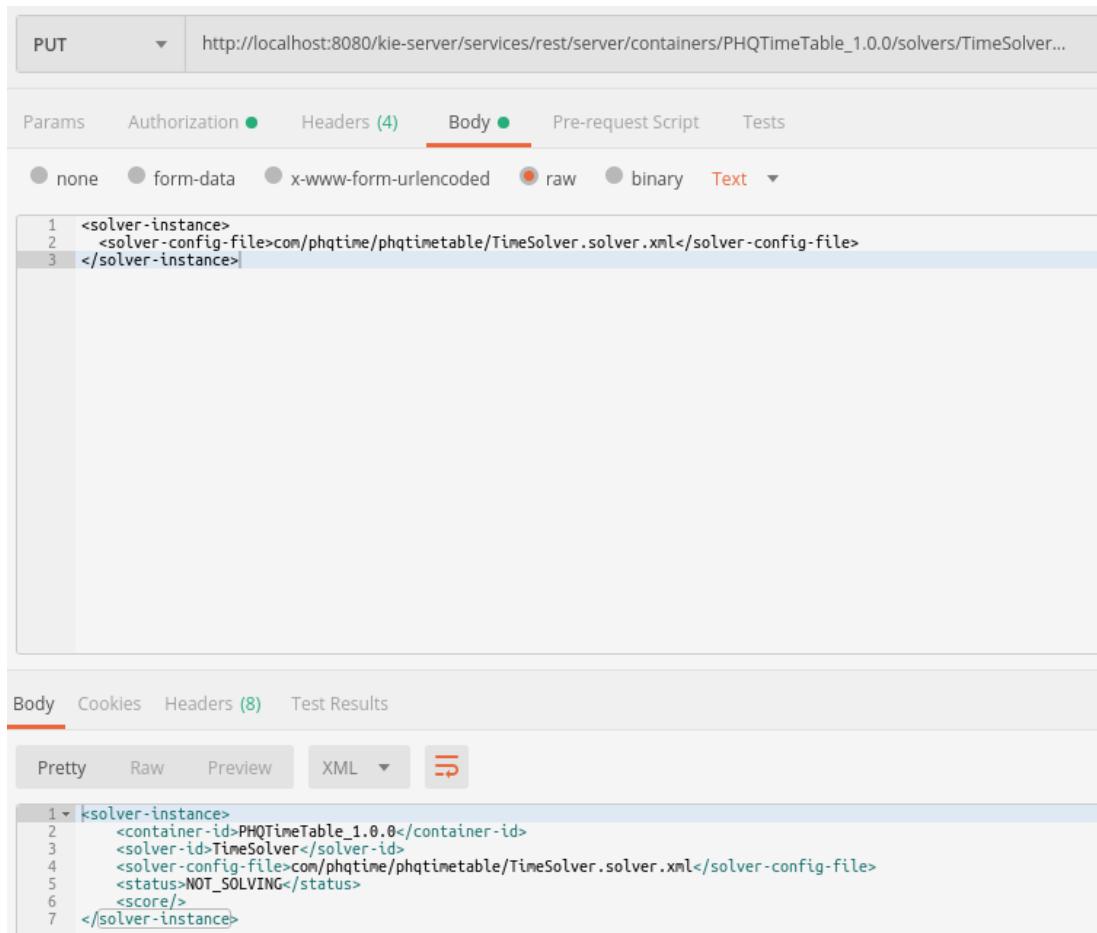
KIE Workbench Setup



Project Implementation

KIE Workbench Setup

Next we use the Tool Postman to test the OptaPlanner application.



Project Implementation

KIE Workbench Setup

When using the construction heuristics, we choose to use First Fit Decreasing. This is to enable us to quickly construct the initial solution.

We started with a very simple search consisting of 3 patients and 3 practitioners, using First Fit Decreasing.

As the State Search Space got larger however, we realize that First Fit Decreasing does not work very well.

http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/state/solving

POST http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/state/solving

Params Authorization Headers (4) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary Text

```
86      <location>central</location>
87    </com.phqtime.phqtimetable.practitioner>
88    <com.phqtime.phqtimetable.practitioner>
89      <id>3</id>
90      <name>Mr Liu Teck Boon</name>
91      <type>psychologist</type>
92      <unavailWeek>5</unavailWeek>
93      <unavail>
94        <com.phqtime.phqtimetable.week_day>
95          <day>saturday</day>
96        </com.phqtime.phqtimetable.week_day>
97        <com.phqtime.phqtimetable.week_day>
98          <day>sunday</day>
99        </com.phqtime.phqtimetable.week_day>
100      </unavail>
101      <language>chinese</language>
102      <gender>male</gender>
103      <cost>100</cost>
104      <location>central</location>
105    </com.phqtime.phqtimetable.practitioner>
106  </practitionerList>
107
108
109  <timeSlotList>
```

Initial testing for First Fit Decreasing with only 3 practitioners and 3 patients

Project Implementation

KIE Workbench Setup

GET

http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/bestsolution

Send

PrettyRawPreviewXML

376<weekday>sunday</weekday>

377</timeSlotList>

378</PHQprocessList>

408</PHQprocessList>

450</PHQprocessList>

451<gender>noPreference</gender>

452<id>3</id>

453<language>chinese</language>

454<location>central</location>

455<name>3</name>

456<PHQscore>26</PHQscore>

457<practitionerSelected>

458<cost>120</cost>

459<gender>male</gender>

460<id>2</id>

461<language>chinese</language>

462<location>central</location>

463<name>Dr Chan Yi Wen Christopher</name>

464<type>psychiatrist</type>

465<unavail>

466<day>monday</day>

467</unavail>

468<unavail>

469<day>tuesday</day>

470</unavail>

471<unavail>

472<day>wednesday</day>

473</unavail>

474<unavail>

475<day>friday</day>

476</unavail>

477<unavail>

478<day>saturday</day>

479</unavail>

480<unavail>

481<day>sunday</day>

482</unavail>

483<unavailWeek>5</unavailWeek>

484</practitionerSelected>

485<preferDay>wednesday</preferDay>

486<remainWeek>1</remainWeek>

487<timeSlotSelected>

488<timeSlot>1000-1100</timeSlot>

489<weekday>thursday</weekday>

490</timeSlotSelected>

491</PHQprocessList>

492<score>69hard/200soft</score>

493</best-solution>

494</solver-instance>

The search outcome for First Fit Decreasing with the small Data Set is successful with a Hard score of 69 and a Soft score of 200.

Project Implementation

KIE Workbench Setup

POST

http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/state/solving

Params

Authorization

Headers (4)

Body

Pre-request Script

Tests

none

form-data

x-www-form-urlencoded

raw

binary

Text

4230

<com.phqtime.phqtimetable.practitioner>

4231

<id>137</id>

4232

<name>Dr Gopal Alakananda</name>

4233

<type>psychiatrist</type>

4234

<unavailWeek>5</unavailWeek>

4235

<unavail>

4236

<com.phqtime.phqtimetable.week_day>

4237

<day>friday</day>

4238

</com.phqtime.phqtimetable.week_day>

4239

<com.phqtime.phqtimetable.week_day>

4240

<day>saturday</day>

4241

</com.phqtime.phqtimetable.week_day>

4242

<com.phqtime.phqtimetable.week_day>

4243

<day>sunday</day>

4244

</com.phqtime.phqtimetable.week_day>

4245

</unavail>

4246

<language>tamil</language>

4247

<gender>male</gender>

4248

<cost>120</cost>

4249

<location>central</location>

4250

</com.phqtime.phqtimetable.practitioner>

4251

</practitionerList>

4252

4253

4254

Next we perform a test on this First Fit

Decreasing search with a bigger data set →

137 practitioners and 134 patients

Project Implementation

KIE Workbench Setup

```
GET http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/bestsolution

Pretty Raw Preview XML

4237 <id>130</id>
4238 <language>tamil</language>
4239 <location>central</location>
4240 <name>130</name>
4241 <PHQscore>12</PHQscore>
4242 <preferDay>monday</preferDay>
4243 <remainWeek>3</remainWeek>
4244 </PHQprocessList>
4245 <PHQprocessList>
4246 <gender>nopreference</gender>
4247 <id>131</id>
4248 <language>english</language>
4249 <location>central</location>
4250 <name>131</name>
4251 <PHQscore>26</PHQscore>
4252 <preferDay>saturday</preferDay>
4253 <remainWeek>2</remainWeek>
4254 </PHQprocessList>
4255 <PHQprocessList>
4256 <gender>nopreference</gender>
4257 <id>132</id>
4258 <language>nopreference</language>
4259 <location>west</location>
4260 <name>90</name>
4261 <PHQscore>10</PHQscore>
4262 <preferDay>monday</preferDay>
4263 <remainWeek>1</remainWeek>
4264 </PHQprocessList>
4265 <PHQprocessList>
4266 <gender>female</gender>
4267 <id>133</id>
4268 <language>nopreference</language>
4269 <location>central</location>
4270 <name>91</name>
4271 <PHQscore>11</PHQscore>
4272 <preferDay>saturday</preferDay>
4273 <remainWeek>3</remainWeek>
4274 </PHQprocessList>
4275 <PHQprocessList>
4276 <gender>nopreference</gender>
4277 <id>134</id>
4278 <language>nopreference</language>
4279 <location>west</location>
4280 <name>134</name>
4281 <PHQscore>12</PHQscore>
4282 <preferDay>wednesday</preferDay>
4283 <remainWeek>1</remainWeek>
4284 </PHQprocessList>
4285 <score>-268init/0hard/0soft</score>
4286 </best-solution>
4287 </solver-instance>
```

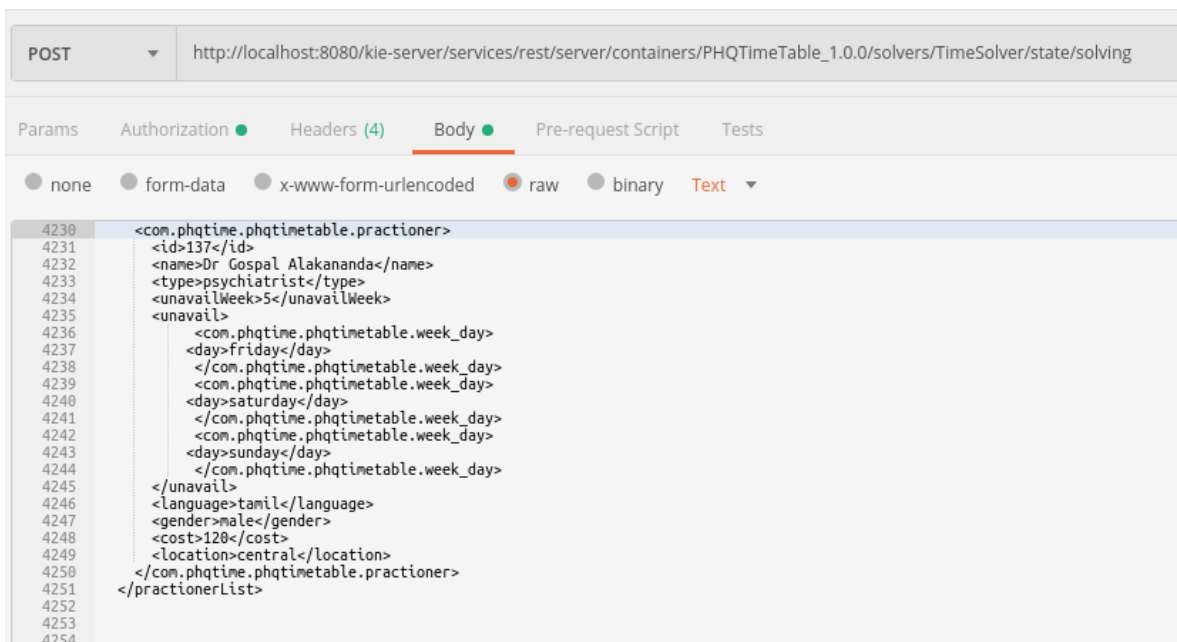
First Fit Decreasing search is fails with this bigger data set of 137 practitioners and 134 patients

Project Implementation

KIE Workbench Setup

Hence, a local search is added, and Late Acceptance search is chosen in this case.

By adding Late Acceptable search after first fit, we aim to generate a solution for 137 practitioners and 134 patients.



```
POST http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/state/solving

<com.phqtime.phqtimetable.practitioner>
  <id>137</id>
  <name>Dr Gopal Alakananda</name>
  <type>psychiatrist</type>
  <unavailWeek>5</unavailWeek>
  <unavail>
    <com.phqtime.phqtimetable.week_day>
      <day>friday</day>
    </com.phqtime.phqtimetable.week_day>
    <com.phqtime.phqtimetable.week_day>
      <day>saturday</day>
    </com.phqtime.phqtimetable.week_day>
    <com.phqtime.phqtimetable.week_day>
      <day>sunday</day>
    </com.phqtime.phqtimetable.week_day>
  </unavail>
  <language>tamil</language>
  <gender>male</gender>
  <cost>120</cost>
  <location>central</location>
</com.phqtime.phqtimetable.practitioner>
</practitionerList>
```


Project Implementation

KIE Workbench Setup

GET

http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/bestsolution

Pretty

Raw

Preview

XML

7223

<location>central</location>

7224

<name>Ms Denise Yap</name>

7225

<type>counsellor</type>

7226

<unavail>

7227

<day>sunday</day>

7228

</unavail>

7229

<unavailWeek>5</unavailWeek>

7230

</practitionerSelected>

7231

<preferDay>saturday</preferDay>

7232

<remainWeek>3</remainWeek>

7233

<timeSlotSelected>

7234

<timeSlot>1000-1100</timeSlot>

7235

<weekday>saturday</weekday>

7236

</timeSlotSelected>

7237

</PHQprocessList>

7238

<PHQprocessList>

7239

<gender>nopreference</gender>

7240

<id>134</id>

7241

<language>nopreference</language>

7242

<location>west</location>

7243

<name>134</name>

7244

<PHQscore>12</PHQscore>

7245

<practitionerSelected>

7246

<cost>100</cost>

7247

<gender>female</gender>

7248

<id>117</id>

7249

<language>tamil</language>

7250

<location>west</location>

7251

<name>Ms Gitanjali Goyal</name>

7252

<type>psychologist</type>

7253

<unavail>

7254

<day>friday</day>

7255

</unavail>

7256

<unavail>

7257

<day>saturday</day>

7258

</unavail>

7259

<unavail>

7260

<day>sunday</day>

7261

</unavail>

7262

<unavailWeek>5</unavailWeek>

7263

</practitionerSelected>

7264

<preferDay>wednesday</preferDay>

7265

<remainWeek>1</remainWeek>

7266

<timeSlotSelected>

7267

<timeSlot>1000-1100</timeSlot>

7268

<weekday>wednesday</weekday>

7269

</timeSlotSelected>

7270

</PHQprocessList>

7271

<score>2485hard/20000soft</score>

7272

</best-solution>

7273

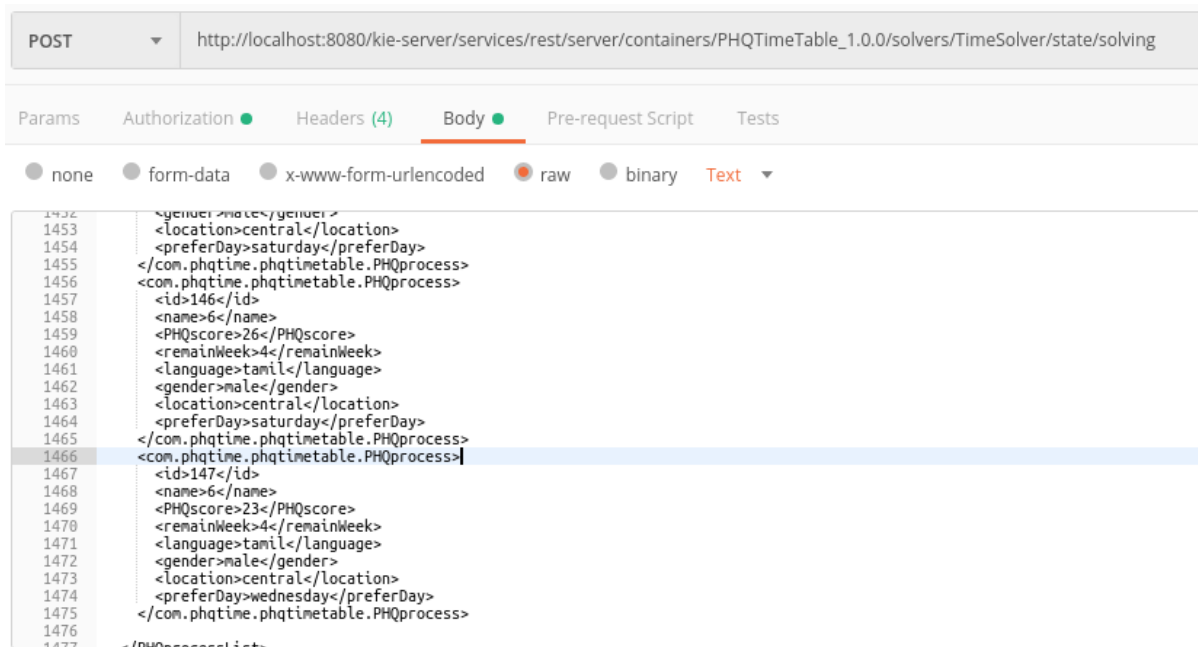
</solver-instance>

Outcome for First Fit
Decreasing with Late
Acceptance search is Positive
with this bigger data set of
137 practitioners and 134
patients

Project Implementation

KIE Workbench Setup

Traditional Hill Climbing has the tendency to be stuck at local optimum but Late Acceptance Hill Climbing increases the chance the to find the global optimum.



Project Implementation

KIE Workbench Setup

GET

http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/bestsolution

Pretty

Raw

Preview

XML

7811

</unavail>

7812

<unavail>

7813

<day>saturday</day>

7814

</unavail>

7815

<unavail>

7816

<day>sunday</day>

7817

</unavail>

7818

<unavailWeek>5</unavailWeek>

7819

</practitionerSelected>

7820

<preferDay>saturday</preferDay>

7821

<remainWeek>4</remainWeek>

7822

<timeSlotSelected>

7823

<timeSlot>0900-1000</timeSlot>

7824

<weekday>saturday</weekday>

7825

</timeSlotSelected>

7826

</PHQprocessList>

7827

<PHQprocessList>

7828

<gender>male</gender>

7829

<id>145</id>

7830

<language>nopreference</language>

7831

<location>central</location>

7832

<name>6</name>

7833

<PHQscore>25</PHQscore>

7834

<preferDay>saturday</preferDay>

7835

<remainWeek>4</remainWeek>

7836

</PHQprocessList>

7837

<PHQprocessList>

7838

<gender>male</gender>

7839

<id>146</id>

7840

<language>tamil</language>

7841

<location>central</location>

7842

<name>6</name>

7843

<PHQscore>26</PHQscore>

7844

<preferDay>saturday</preferDay>

7845

<remainWeek>4</remainWeek>

7846

</PHQprocessList>

7847

<PHQprocessList>

7848

<gender>male</gender>

7849

<id>147</id>

7850

<language>tamil</language>

7851

<location>central</location>

7852

<name>6</name>

7853

<PHQscore>23</PHQscore>

7854

<preferDay>wednesday</preferDay>

7855

<remainWeek>4</remainWeek>

7856

</PHQprocessList>

7857

<score>-6init/2714hard/23760soft</score>

7858

</best-solution>

7859

</solver-instance>

Result showing that Hill Climbing getting stuck at local optimum.

Project Implementation

KIE Workbench Setup

GET

http://localhost:8080/kie-server/services/rest/server/containers/PHQTimeTable_1.0.0/solvers/TimeSolver/bestsolution

Pretty

Raw

Preview

XML

7871

</unavail>

7872

<unavail>

7873

<day>sunday</day>

7874

</unavail>

7875

<unavailWeek>5</unavailWeek>

7876

</practionerSelected>

7877

<preferDay>saturday</preferDay>

7878

<remainWeek>4</remainWeek>

7879

<timeSlotSelected>

7880

<timeSlot>2000-2100</timeSlot>

7881

<weekday>monday</weekday>

7882

</timeSlotSelected>

7883

</PHQprocessList>

7884

<PHQprocessList>

7885

<gender>male</gender>

7886

<id>147</id>

7887

<language>tamil</language>

7888

<location>central</location>

7889

<name>6</name>

7890

<PHQscore>23</PHQscore>

7891

<practionerSelected>

7892

<cost>120</cost>

7893

<gender>male</gender>

7894

<id>137</id>

7895

<language>tamil</language>

7896

<location>central</location>

7897

<name>Dr Gopal Alakananda</name>

7898

<type>psychiatrist</type>

7899

<unavail>

7900

<day>friday</day>

7901

</unavail>

7902

<unavail>

7903

<day>saturday</day>

7904

</unavail>

7905

<unavail>

7906

<day>sunday</day>

7907

</unavail>

7908

<unavailWeek>5</unavailWeek>

7909

</practionerSelected>

7910

<preferDay>thursday</preferDay>

7911

<remainWeek>4</remainWeek>

7912

<timeSlotSelected>

7913

<timeSlot>0900-1000</timeSlot>

7914

<weekday>thursday</weekday>

7915

</timeSlotSelected>

7916

</PHQprocessList>

7917

<score>2816hard/24210soft</score>

7918

</best-solution>

7919

</solver-instance>

Result showing Late
Acceptance Hill Climbing
solving the problem.

Project Implementation

Search Design

From our earlier tests, it is observed that the First Fit algorithm provides a fast but often non-optimal solution. The algorithm can be made much more effective by first sorting the list of elements into decreasing order, as in First Fit Decreasing.

We learnt that some optimization algorithms work more efficiently when we identify which planning entities are more difficult to plan. In our case, these planning entities are practitioner and work period.

Our team has created some Java classes (working in tandem with Optaplaner) to execute based on First Fit Decreasing. Following shows the Java class encapsulating both planning variables – for practitioner and for work period.

```
@PlanningEntity(difficultyWeightFactoryClass = PhqProcessDifficultyWeightFactory.class)
@XStreamAlias("PhqProcess")
public class PhqProcess implements java.io.Serializable {

    static final long serialVersionUID = 1L;


    private long id;

    @PlanningVariable(valueRangeProviderRefs = { "practionerRange" },
        strengthWeightFactoryClass = PhqPractionerStrengthWeightFactory.class)
    private PhqPractioner selectedPractioner;

    @PlanningVariable(valueRangeProviderRefs = { "periodRange" },
        strengthWeightFactoryClass = WorkPeriodStrengthWeightFactory.class)
    private WorkPeriod selectedPeriod;

    private PhqPatient assignedPatient;

    @PlanningPin
    private boolean planningPinStatus;
```



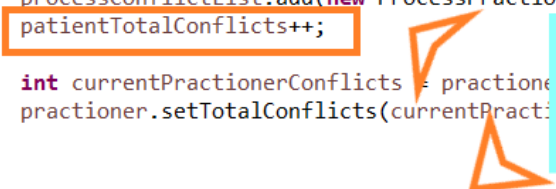
Define the strength weight factory class to control the search selection sequence

Project Implementation

Search Design

The following logic helps to ensure that we match those patients with the least number of assignable practitioners first as they are more difficult to match.

```
if (conflictCount > 0) {  
    processConflictList.add(new ProcessPractitionerConflict(process, practitioner, conflictCount));  
    patientTotalConflicts++;  
    int currentPractitionerConflicts = practitioner.getTotalConflicts();  
    practitioner.setTotalConflicts(currentPractitionerConflicts + conflictCount);  
}  
patient.setTotalConflicts(patientTotalConflicts);
```



Inside PhqSessionScheduleSolution class method
prepareConflictList(), define the patient conflict
based on how many practitioner can not be assigned to the
patient

Project Implementation

Search Design

The following serves to match those patients with more constraints first.

```
public class PhqProcessDifficultyWeightFactory implements SelectionSorterWeightFactory<PhqSessionScheduleSolution, PhqProcess> {  
    public PhqProcessDifficultyWeight createSorterWeight(PhqSessionScheduleSolution schedule, PhqProcess process) {  
        PhqPatient patient = process.getAssignedPatient();  
        int patientDifficultyCount = 0;  
        if (patient.getPreferLanguage() != patient.getPreferLanguageIndex("English")) {  
            patientDifficultyCount++;  
        }  
        if (patient.getPreferGender() != patient.getPreferGenderIndex("Any")) {  
            patientDifficultyCount++;  
        }  
        if (patient.getPreferLocation() != patient.getPreferLocationIndex("Any")) {  
            patientDifficultyCount++;  
        }  
        return new PhqProcessDifficultyWeight(process, patientDifficultyCount);  
    }  
}  
  
public static class PhqProcessDifficultyWeight implements Comparable<PhqProcessDifficultyWeight> {  
    private final PhqProcess process;  
    private final int processDifficultyCount;  
  
    public PhqProcessDifficultyWeight(PhqProcess process, int processDifficultyCount) {  
        this.process = process;  
        this.processDifficultyCount = processDifficultyCount;  
    }  
  
    public int compareTo(PhqProcessDifficultyWeight other) {  
        return new CompareToBuilder()  
            // Processing patient with more conflict first  
            .append(other.process.getAssignedPatient().getTotalConflicts(), process.getAssignedPatient().getTotalConflicts())  
            // Processing difficult patient assignment first  
            .append(other.processDifficultyCount, processDifficultyCount)  
            .append(process.getId(), other.process.getId())  
            .toComparison();  
    }  
}
```

Check preferred language

Check preferred gender

Check preferred location

Assign the process with the patient having more conflict on practioner

Assign the process with the patient having more difficulties to locate practioner

Project Implementation

Search Design

Here, patients are assigned slots starting from beginning of week and from earlier time of day onwards.

```
public class WorkPeriodStrengthWeightFactory implements SelectionSorterWeightFactory<PhqSessionScheduleSolution, WorkPeriod> {  
  
    public WorkPeriodStrengthWeight createSorterWeight(PhqSessionScheduleSolution schedule, WorkPeriod period) {  
        List<WorkPeriod> notFreePeriodList = schedule.getUnavailablePeriodList();  
        int periodDifficultyCount = 0;  
        for (WorkPeriod notFreePeriod : notFreePeriodList) {  
            if (notFreePeriod == period) {  
                periodDifficultyCount++;  
            }  
        }  
  
        return new WorkPeriodStrengthWeight(period, periodDifficultyCount);  
    }  
  
    public static class WorkPeriodStrengthWeight implements Comparable<WorkPeriodStrengthWeight> {  
  
        private final WorkPeriod period;  
        private final int periodStrengthCount;  
  
        public WorkPeriodStrengthWeight(WorkPeriod period, int periodStrengthCount) {  
            this.period = period;  
            this.periodStrengthCount = periodStrengthCount;  
        }  
  
        public int compareTo(WorkPeriodStrengthWeight other) {  
            return new CompareToBuilder()  
                // Processing less busy day first  
                .append(periodStrengthCount, other.periodStrengthCount)  
                // Assignment  
                .append(period.getId(), other.period.getId())  
                .toComparison();  
        }  
    }  
}
```

Check how many practioners are not available at the given period

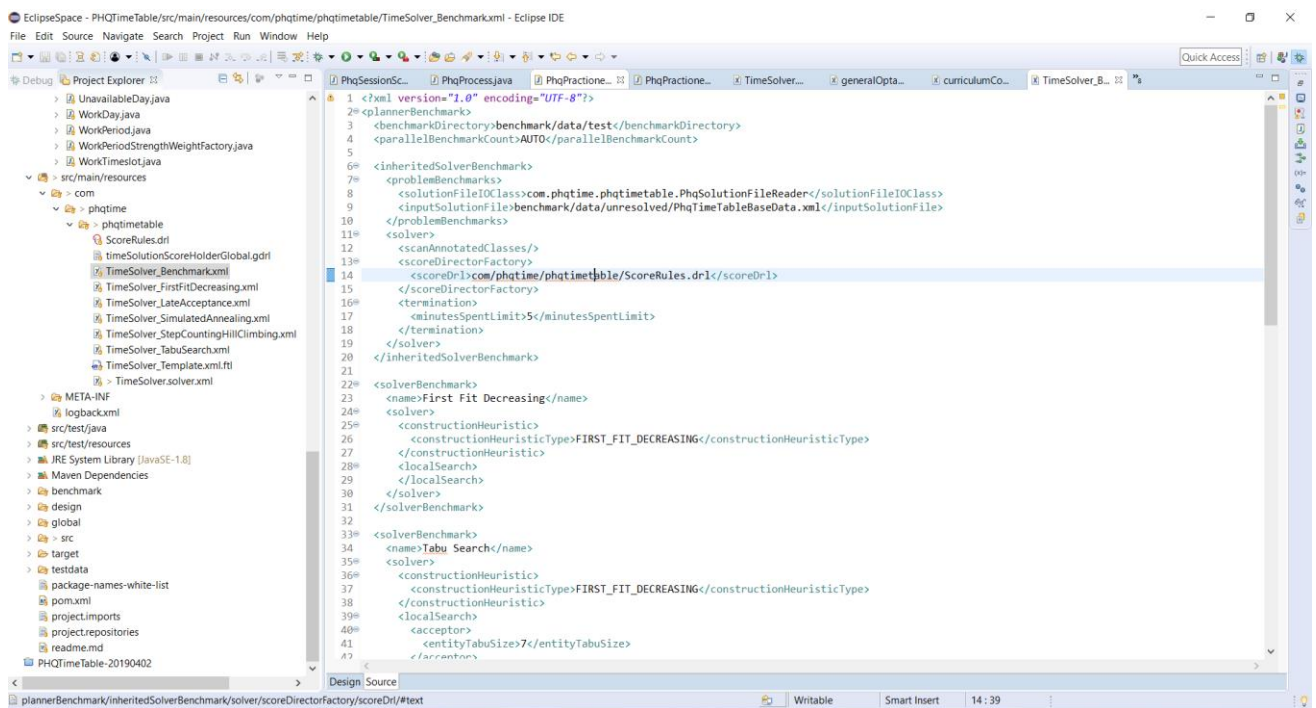
Assign the period on which has more available practioners

Allocate period in sequence such that from Mon to Sun, and from 8:00AM to 4:00PM

Project Performance & Validation

Benchmark

In the Eclipse environment, we have created an XML program to test the various search algorithms. This serves to benchmark the performance of each search method which is outlined on the next page.

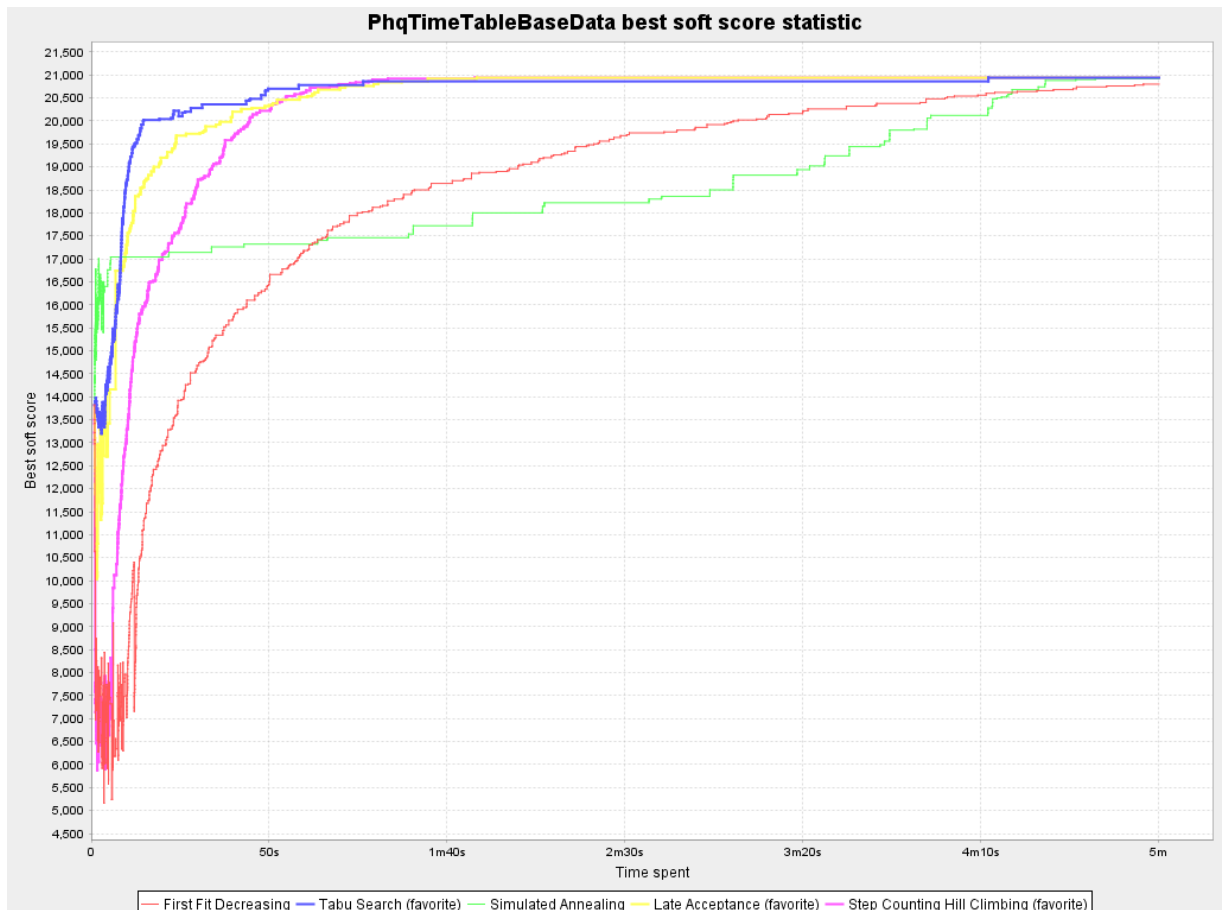


Project Performance & Validation

Benchmark

A benchmark test was conducted against the various search algorithms as follows. We have made some observations as follows. All hard constraints are met. The level of soft constraints being met varies across the different search algorithms.

- 1) First Fit Decreasing
This is the slowest search algorithm in our scenario. This demonstrates a key characteristic of construction heuristic vs metaheuristic.
- 2) Tabu Search
This search made the fastest progress in the beginning but took a long time to converge into global optimum compared to Late Acceptance.
- 3) Simulated Annealing
This is the slowest search.
- 4) Late Acceptance
This is the fastest to reach the global optimum.
- 5) Step Counting Hill Climbing
This has reasonable performance and reaches global optimum quite fast.



Project Conclusions: Findings & Recommendation

We have met the prescribed objective of this project and that is to optimally match a group of patients with a group of practitioners, within boundary of hard constraints and with optimal compliance to soft constraints. In addition, we have employed the different search algorithms to understand the characteristics of each of them.

Hence, we consider this application to be more than meeting the objectives and ready to be adapted for a real world requirement.

More importantly, we have gained some good knowledge concerning the various search methodologies and optimization techniques.

Starting with the First Fit Decreasing, we have learnt that this forms the backbone of search methodologies because it assigns the more difficult planning entities first. These difficult planning entities are less likely to fit in the leftovers. So, sorting the planning entities on decreasing difficulty enables some efficiency in the search execution.

However, algorithms like First Fit Decreasing can lead to the search getting stuck at local optimums. This is where metaheuristic algorithms come in. In fact, in our case, the Late Acceptance algorithm fares best. Other algorithms have varying characteristics depending on specific settings. One noteworthy observation is that Tabu Search enabled a quick trajectory towards global optimum but took some time to reach it.