Red Hat Process Automation Manager 7.7

Red Hat Business Optimizer のスタートガイド

ガイド

Last Updated: 2021-08-02

Red Hat Process Automation Manager 7.7 Red Hat Business Optimizer のスタートガイド

ガイド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、Red Hat Business Optimizer をどのように使用開始するか説明しています。

目次

前書き

ビジネスルールの開発者は、Red Hat Business Optimizer を使用して、限られたリソースや個別の制約の中で計画問題に対する最適解を見つけ出すことができます。

本書を使用して、Red Hat Business Optimizer で Solver の開発を開始していきます。

第1章 RED HAT BUSINESS OPTIMIZER の概要

Red Hat Business Optimizer は組み込み可能な軽量プランニングエンジンで、プランニングの問題を最適化します。最適化のためのヒューリスティック法およびメタヒューリスティック法を、非常に効率的なスコア計算と組み合わせ、一般的な Java プログラマーが計画問題を効率的に解決できるようにします。

Red Hat Business Optimizer は、さまざまなユースケースの解決に役立ちます。以下にその例を示します。

- 従業員勤務表/患者一覧: 看護師の勤務シフト作成を容易にし、病床管理を追跡します。
- 教育機関の時間割: 授業、コース、試験、および会議の計画を容易にします。
- 工場の計画: 自動車の組み立てライン、機械の操業計画、および作業員のタスク計画を追跡します。
- 在庫の削減: 紙や金属などの資源の消費を減らし、無駄を最小限に抑えます。

どの組織も、制約のある限定されたリソース (従業員、資産、時間、および資金) を使用して製品やサービスを提供するといった計画問題に直面しています。

Red Hat Business Optimizer は、Apache Software License 2.0 を使用するオープンソースソフトウェアです。100% Pure Java に認定されており、ほとんどの Java 仮想マシンで稼働します。

1.1. 計画問題

計画問題 では、限られたリソースや個別の制約の中で最適なゴールを見つけ出します。最適なゴールは、次のようなさまざまなものです。

- 最大の利益: 最適なゴールにより、可能な限り高い利益が得られます。
- 経済活動の最小フットプリント: 最適なゴールでは、環境負荷が最小となります。
- スタッフ/顧客の最大満足: 最適なゴールでは、スタッフ/顧客のニーズが優先されます。

これらのゴールに到達できるかどうかは、利用できるリソースの数に依存します。たとえば、以下のようなリソースには制限があります。

- 要員の人数
- 時間
- 予算
- 装置、車両、コンピューター、施設などの物理資産

これらのリソースに関連する個別の制約についても考慮する必要があります。たとえば、要員が働くことのできる時間数、特定の装置を使用することのできる技能、または機器同士の互換性などです。

Red Hat Business Optimizer は、Java プログラマーが制約の飽和性の問題を効率的に解決するのに役立ちます。最適化ヒューリスティックとメタヒューリスティックを効率的なスコア計算と組み合わせます。

1.2. 計画問題での NP 完全

例に挙げたユースケースは 通常 NP 完全または NP 困難 であり、以下のことが言えます。

- 問題に対する解を実用的な時間内に検証することが容易である。
- 問題に対する最適解を実用的な時間内に見つけ出す確実な方法がない。

この場合、一般的な 2 つの手法では不十分であるため、問題を解くのが予想より困難だと考えられます。

- 力まかせアルゴリズムでは (より高度な類似アルゴリズムであっても)、時間がかかり過ぎる。
- (「ビンパッキング問題」において) 容量が一番大きいビンから順に詰めてゆく クイックアルゴリズムでは、最適解からは程遠い解しか得られない。

高度な最適化アルゴリズムを用いる Business Optimizer であれば、このような計画問題に対する適切な解を、妥当な時間内に見つけ出すことができます。

1.3. 計画問題に対する解

計画問題には、多数の解が存在します。

以下に示すように、解は複数のカテゴリーに分類されます。

可能解
: 可能解とは、制約に違反するかどうかは問わず、あらゆる解を指します。通常、計画問題には膨大な数の可能解が存在します。ただし、このような解の多くは、役に立ちません。

実行可能解
: 実行可能解とは、いずれの (負の) ハード制約にも違反しない解を指します。実行可能解の数は、可能解の数に比例します。実行可能解が存在しないケースもあります。実行可能解は、可能解の部分集合です。

最適解
: 最適解とは、最高のスコアを持つ解を指します。通常、計画問題には数個の最適解が存在します。実行可能解が存在せず、最適解が現実的ではない場合でも、計画問題には少なくとも 1 つの最適解が必ず存在します。

見つかった最善解
: 最善解とは、与えられた時間内に実施した検索で見つかった最高スコアの解を指します。通常、見つかった最善解は現実的で、十分な時間があれば最適解を見つけることができます。

直観には反していますが、小規模なデータセットの場合であっても、(正しく計算された場合は) 膨大な数の可能解が存在します。

planner-engine ディストリビューションディレクトリーのサンプルでも、ほとんどのインスタンスには多数の可能解が存在します。最適解を確実に見つけることができる方法は存在しないため、いかなる実行方法も、これらすべての可能解の部分集合を評価することしかできません。

膨大な数の可能解全体を効率的に網羅するために、Business Optimizer はさまざまな最適化アルゴリズムをサポートしています。

ユースケースによっては、ある最適化アルゴリズムが他のアルゴリズムより勝ることがありますが、それを事前に予測することは不可能です。Business Optimizer では、XML またはコード中の Solver 設定を数行変更するだけで、最適化アルゴリズムを切り替えることができます。

1.4. 計画問題に対する制約

通常、プランニングの問題には、少なくとも 2 つの制約レベルがあります。

- (負の) ハード制約 は、絶対に違反してはならない。
  例: 1 人の教師は同時に 2 つの講義を受け持つことはできない。

- (負の) ソフト制約 は、避けることが可能であれば違反してはならない。
  例: 教師 A は金曜日の午後に講義を受け持ちたくない。

正の制約を持つ問題もあります。

- 正のソフト制約 (ボーナス) は、可能であれば満たす必要がある。
  例: 教師 B は月曜日の午前中に講義を受け持つことを希望している。

一部の基本的な問題にはハード制約のみがあります。問題によっては、3 つ以上の制約があります (例: ハード制約、中程度の制約、ソフト制約)。

これらの制約により、計画問題における スコア計算方法 (または 適合度関数) が定義されます。プランニングの問題の解は、それぞれスコアで等級付けすることができます。Business Optimizer のスコア制約は、Java などのオブジェクト指向言語または Drools ルールで記述されます。

このタイプのコードは柔軟で、スケーラビリティーに優れます。

第2章 BUSINESS CENTRAL での SOLVER のスタートガイド: 従業員勤務表の例

Business Central で **employee-rostering** のサンプルプロジェクトを構築して、デプロイできます。このプロジェクトは、シフト勤務の計画問題を解決するのに必要な Business Central の各アセットを作成し、Red Hat Business Optimizer を使用して実現可能な最適解を見つける方法を示します。

Business Central に事前設定された **employee-rostering** プロジェクトをデプロイすることができます。Business Central を使用してプロジェクトを独自に作成することができます。

注記

Business Central の **employee-rostering** サンプルプロジェクトには、データセットが含まれていません。REST API 呼び出しを使用して XML 形式のデータセットを提供する必要があります。

2.1. BUSINESS CENTRAL への従業員勤務表サンプルプロジェクトのデプロイメント

Business Central には、製品と機能に慣れるために使用できるサンプルプロジェクトが多数あります。従業員の勤務表サンプルプロジェクトは、Red Hat Business Optimizer でシフト勤務のユースケースを示すために計画され作成されました。以下の手順に従って、従業員の勤務表サンプルを Business Central にデプロイして実行します。

前提条件

- Red Hat Process Automation Manager をダウンロードして、インストールしている。インストールオプションは『Red**Hat Process Automation Manager** インストールの計画』を参照してください。
- インストールドキュメント の説明に従って Red Hat Process Automation Manager を起動し、**admin** パーミッションを持つユーザーとして Business Central にログインしている。

手順

1. Business Central で **Menu** → **Design** → **Projects** の順にクリックします。
2. 事前に設定した **MySpace** スペースで **Try Samples** をクリックします。
3. サンプルプロジェクトの一覧から **employee-rostering** を選択し、右上の **OK** をクリックして、プロジェクトをインポートします。
4. アセットリストをコンパイルし、**Build & Deploy**をクリックして、従業員の勤務表サンプルをデプロイします。

本書は、各プロジェクトアセットとその設定について説明します。

2.2. 従業員の勤務表サンプルプロジェクトの再作成

従業員の勤務表サンプルプロジェクトは、Business Central で使用できる事前設定のプロジェクトです。このプロジェクトをデプロイする方法は「 「Business Central への従業員勤務表サンプルプロジェクトのデプロイメント」 」を参照してください。

「ゼロから」従業員勤務表を作成できます。この例では、ワークフローを使用して、Business Central で独自の類似プロジェクトを作成します。

2.2.1. 従業員の勤務表プロジェクトの設定

Business Central で Solver の開発を始めるには、プロジェクトを設定する必要があります。

前提条件

- Red Hat Process Automation Manager をダウンロードして、インストールしている。
- Business Central をデプロイし、**admin** ロールを持つユーザーでログインしている。

手順

1. **Menu → Design → Projects → Add Project** をクリックして、Business Central に新しいプロジェクトを作成します。
2. **Add Project** ウィンドウで、以下のフィールドに入力します。
   - Name: **employee-rostering**
   - **Description** (任意): Business Optimizer を使用した従業員の勤務表問題の最適化。スキルに基づいて、従業員をシフトに割り当てます。

   任意で、**Configure Advanced Options** をクリックして、**Group ID**、**Artifact ID**、および **Version** に情報を追加します。
   - Group ID: **employeerostering**
   - Artifact ID: **employeerostering**
   - Version: **1.0.0-SNAPSHOT**
3. **Add** をクリックして、Business Central プロジェクトリポジトリーにプロジェクトを追加します。

2.2.2. プロジェクトファクトおよびプランニングエンティティー

従業員勤務表の計画問題の各ドメインクラスは、以下のいずれかに分類されます。

- 関連性のないクラス: どのスコア制約にも使用されません。計画に関して言えば、このデータは使用されません。
- 問題ファクト クラス: スコア制約に使用されますが、(問題が変わらない限り) 計画時には変化しません (例: **Shift**、**Employee**)。問題ファクトクラスのプロパティーはすべて問題のプロパティーです。
- プランニングエンティティークラス クラス: スコア制約に使用され、計画時に変化します (例: **ShiftAssignment**)。計画時に変更するプロパティーは プランニング変数 です。その他のプロパティーは問題プロパティーです。
  以下の点についてお考え下さい。
- 計画時にどのクラスを変更しますか?
- **Solver** で変更する変数はどのクラスにありますか?

そのクラスが、プランニングエンティティーです。

プランニングエンティティークラスは、**@PlanningEntity** アノテーションでアノテートする必要があります。または、ドメインデザイナーで Red Hat Business Optimizer ドックを使用して Business Central に定義する必要があります。

各プランニングエンティティークラスには、1 つ以上の プランニング変数 があり、1 つ以上の定義プロパティーが必要です。

多くのユースケースには、プランニングエンティティークラスが 1 つだけあり、1 つのプランニングエンティティークラスに対してプランニング変数が 1 つだけ含まれます。

2.2.3. 従業員の勤務表プロジェクトへのデータモデルの作成

このセクションでは、Business Central で従業員の勤務表サンプルプロジェクトを実行するのに必要なデータオブジェクトを作成します。

前提条件

- 「従業員の勤務表プロジェクトの設定」 に従ってプロジェクト設定が完了している。

手順

1. 新規プロジェクトで、プロジェクトパースペクティブの **Data Object** をクリックするか、**Add Asset → Data Object** をクリックして、新しいデータオブジェクトを作成します。
2. 最初のデータオブジェクトの名前を **Timeslot** とし、パッケージ で **employeerostering.employeerostering** を選択します。
   OK をクリックします。
3. **Data Objects** パースペクティブで **+add field** をクリックして、**Timeslot** データオブジェクトにフィールドを追加します。
4. **id** フィールドで **endTime** と入力します。
5. **Type** の横にあるドロップダウンメニューをクリックし、**LocalDateTime** を選択します。
6. **Create and continue** を別のフィールドに追加します。
7. **id startTime** および **Type LocalDateTime** を使用して、フィールドを追加します。
8. **Create** をクリックします。
9. 右上の **Save** をクリックして、**Timeslot** データオブジェクトを保存します。
10. 右上の **x** をクリックして、**Data Objects** パースペクティブを閉じ、**Assets** メニューに戻ります。
11. 前述の手順で、以下のデータオブジェクトとその属性を作成します。

表2.1 Skill

id	タイプ
name	**String**

表2.2 Employee

id	タイプ
name	**String**
skills	**employeerostering.employeerostering.Skill[List]**

表2.3 Shift

id	タイプ
requiredSkill	**employeerostering.employeerostering.Skill**
timeslot	**employeerostering.employeerostering.Timeslot**

表2.4 DayOffRequest

id	タイプ
date	**LocalDate**
employee	**employeerostering.employeerostering.Employee**

表2.5 ShiftAssignment

id	タイプ
employee	**employeerostering.employeerostering.Employee**
shift	**employeerostering.employeerostering.Shift**

データオブジェクトの作成例は『デシジョンサービスの使用ガイド』を参照してください。

2.2.3.1. 従業員の勤務表プランニングエンティティーの作成

従業員勤務表の計画問題を解決するには、プランニングエンティティーと Solver を作成する必要があります。プランニングエンティティーは、Red Hat Business Optimizer ドックで利用可能な属性を使用して、ドメインデザイナーに定義します。

以下の手順に従って、従業員の勤務表サンプルに、**ShiftAssignment** データオブジェクトをプランニングエンティティーとして定義します。

前提条件

- 従業員の勤務表サンプルを実行するには、「「従業員の勤務表プロジェクトへのデータモデルの作成」」の手順に従って、関連するデータオブジェクトとプランニングエンティティーを作成する必要があります。

手順

1. プロジェクトの **Assets** メニューから、**ShiftAssignment** データオブジェクトを開きます。
2. **Data Objects** パースペクティブで、右側の  をクリックして、Red Hat Business Optimizer ドックを開きます。
3. **Planning Entity** を選択します。
4. **ShiftAssignment** データオブジェクトのフィールドリストで **employee** を選択します。
5. Red Hat Business Optimizer ドックで **Planning Variable** を選択します。**Value Range Id** 入力フィールドに **employeeRange** を入力します。これにより、**@ValueRangeProvider** アノテーションがプランニングエンティティーに追加され、デザイナーの **Source** タブをクリックすると表示されます。

   プランニング変数の値の範囲は **@ValueRangeProvider** アノテーションで定義されます。**@ValueRangeProvider** アノテーションには **id** プロパティーが常にあり、**@PlanningVariable** の **valueRangeProviderRefs** プロパティーから参照されます。
6. ドックを閉じ、**Save** をクリックして、データオブジェクトを保存します。

2.2.3.2. 従業員の勤務表プランニングソリューションの作成

従業員勤務表の問題は、定義したプランニングソリューションに依存します。プランニングソリューションは、Red Hat Business Optimizer ドックで利用可能な属性を使用してドメインデザイナーで定義されます。

前提条件

- 「「従業員の勤務表プロジェクトへのデータモデルの作成」」および「従業員の勤務表プランニングエンティティーの作成」の手順に従って、従業員の勤務表サンプルを実行するのに必要なデータオブジェクトおよびプランニングエンティティーを作成している。

手順

1. 識別子 **EmployeeRoster** でデータオブジェクトを新規作成します。
2. 以下のフィールドを作成します。

   表2.6 EmployeeRoster

id	タイプ
dayOffRequestList	**employeerostering.employeerostering.DayOffRequest[List]**

id	タイプ
shiftAssignmentList	**employeerostering.employeerostering.ShiftAssignment[List]**
shiftList	**employeerostering.employeerostering.Shift[List]**
skillList	**employeerostering.employeerostering.Skill[List]**
timeslotList	**employeerostering.employeerostering.Timeslot[List]**

3. **Data Objects** パースペクティブで、右側の  をクリックして、Red Hat Business Optimizer ドックを開きます。
4. **Planning Solution** を選択します。
5. **Solution Score Type** は、デフォルトの **Hard soft score** のままにします。これにより、タイプがソリューションスコアとなる **EmployeeRoster** データオブジェクトに、**score** フィールドが自動的に生成されます。
6. 次の属性で新しいフィールドを追加します。

id	タイプ
employeeList	**employeerostering.employeerostering.Employee[List]**

7. **employeeList** フィールドを選択した状態で、Red Hat Business Optimizer ドックを開いて、**Planning Value Range Provider** ボックスを選択します。
id フィールドに **employeeRange** を入力します。ドックを閉じます。
8. 右上で **Save** をクリックし、アセットを保存します。

2.2.4. 従業員勤務表の制約

従業員の勤務表はプランニングソリューションです。すべての計画問題には、最適解を得るのに満たさなければならない制約が含まれます。

Business Central の従業員の勤務表サンプルプロジェクトには、以下のハード制約およびソフト制約が含まれます。

ハード制約

- 従業員に割り当てられるシフトの数は、1日1つまで。
- 特別な従業員スキルが必要なすべてのシフトは、そのスキルを持つ従業員に割り当てられる。

ソフト制約

- すべての従業員がシフトに割り当てられている。
- 従業員が休暇を取った場合は、シフトが別の従業員に再割り当てされる。

ハード制約およびソフト制約は、Free form DRL デザイナー、またはガイド付きルールを使用して Business Central で定義します。

2.2.4.1. DRL (Drools Rule Language) ルール

DRL (Drools Rule Language) ルールは、**.drl** テキストファイルに直接定義するビジネスルールです。このような DRL ファイルは、Business Central の他のすべてのルールアセットが最終的にレンダリングされるソースとなります。Business Central インターフェースで DRL ファイルを作成して管理するか、Red Hat CodeReady Studio や別の統合開発環境 (IDE) を使用して Maven または Java プロジェクトの一部として外部で作成することができます。DRL ファイルには、最低でもルールの条件 (**when**) およびアクション (**then**) を定義するルールを 1 つ以上追加できます。Business Central の DRL デザイナーでは、Java、DRL、および XML の構文が強調表示されます。

DRL ファイルは、以下のコンポーネントで構成されます。

DRL ファイル内のコンポーネント

```
package

import

function  // Optional

query  // Optional

declare   // Optional

global   // Optional

rule "rule name"
    // Attributes
    when
        // Conditions
    then
        // Actions
end

rule "rule2 name"

...
```

以下の DRL ルールの例では、ローン申し込みのデシジョンサービスで年齢制限を指定します。

申込者の年齢制限に関するルールの例

```
rule "Underage"
  salience 15
  agenda-group "applicationGroup"
```

```
  when
    $application : LoanApplication()
    Applicant( age < 21 )
  then
    $application.setApproved( false );
    $application.setExplanation( "Underage" );
end
```

DRL ファイルには、ルール、クエリー、関数が 1 つまたは複数含まれており、このファイルで、ルールやクエリーで割り当て、使用するインポート、グローバル、属性などのリソース宣言を定義できます。DRL パッケージは、DRL ファイルの一番上に表示され、ルールは通常最後に表示されます。他の DRL コンポーネントはどのような順番でも構いません。

ルールごとに、ルールパッケージ内で一意の名前を指定する必要があります。パッケージ内の DRL ファイルで、同じルール名を複数回使用すると、ルールのコンパイルに失敗します。特にルール名にスペースを使用する場合など、ルール名には必ず二重引用符 (**rule "rule name"**) を使用して、コンパイルエラーが発生しないようにしてください。

DRL ルールに関連するデータオブジェクトはすべて、DRL ファイルと同じ Business Central プロジェクトパッケージに置く必要があります。同じパッケージに含まれるアセットはデフォルトでインポートされます。その他のパッケージの既存アセットは、DRL ルールを使用してインポートできます。

2.2.4.2. DRL デザイナーを使用した従業員の勤務表の制約定義

Business Central で Free form DRL デザイナーを使用して、従業員の勤務表サンプルに制約の定義を作成できます。

この手順を使用して、シフトが終わってから 10 時間以上経たないと従業員をシフトに割り当てられない ハード制約 を作成します。

手順

1. Business Central で、**Menu → Design → Projects** に移動して、プロジェクト名をクリックします。
2. **Add Asset → DRL** ファイル の順にクリックします。
3. **DRL file** 名前フィールドに、**ComplexScoreRules** と入力します。
4. **employeerostering.employeerostering** パッケージを選択します。
5. **+OK** をクリックして DRL ファイルを作成します。
6. DRL デザイナーの **Model** タブで、**Employee10HourShiftSpace** ルールを DRL ファイルとして定義します。

```
package employeerostering.employeerostering;

rule "Employee10HourShiftSpace"
    when
        $shiftAssignment : ShiftAssignment( $employee : employee != null, $shiftEndDateTime :
shift.timeslot.endTime)
        ShiftAssignment( this != $shiftAssignment, $employee == employee, $shiftEndDateTime
<= shift.timeslot.endTime,
                $shiftEndDateTime.until(shift.timeslot.startTime,
java.time.temporal.ChronoUnit.HOURS) <10)
```

```
    then
        scoreHolder.addHardConstraintMatch(kcontext, -1);
end
```

7. **Save** をクリックして、DRL ファイルを保存します。

DRL ファイルの作成方法は『DRLルールを使用したデシジョンサービスの作成』を参照してください。

2.2.5. ガイド付きルールを使用して従業員の勤務表にルールの作成

Business Central でガイド付きルールデザイナーを使用して、従業員の勤務表にハード制約およびソフト制約を定義するルールを作成できます。

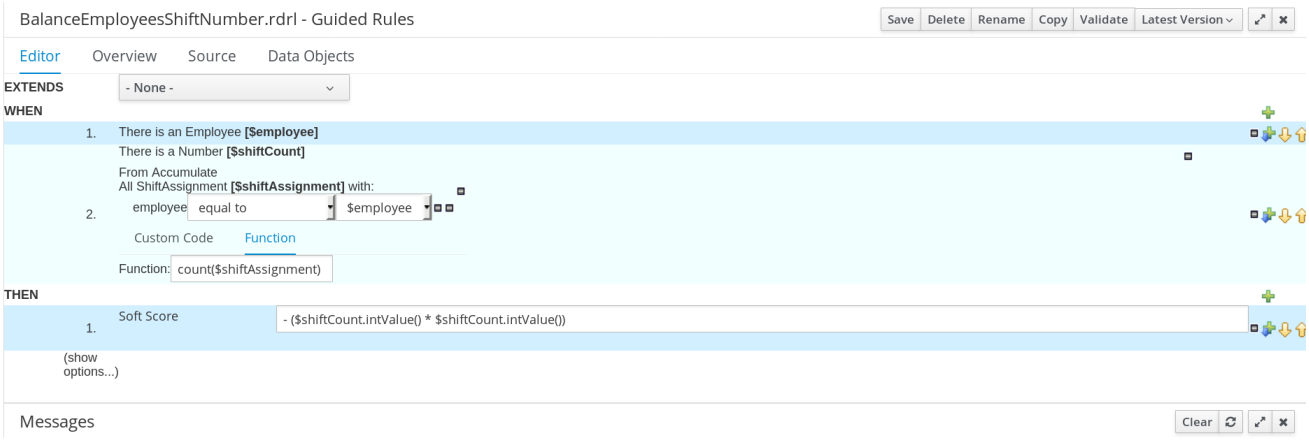
2.2.5.1. ガイド付きルール

ガイド付きルールは、ルール作成のプロセスを提供する、Business Central の UI ベースのガイド付きルールデザイナーで作成するビジネスルールです。ガイド付きルールデザイナーを使用すると、ルールを定義するデータオブジェクトに基づいて、可能なインプットにフィールドおよびオプションを提供します。定義したガイド付きルールは、その他のすべてのルールアセットとともに Drools Rule Language (DRL) ルールにコンパイルされます。

ガイド付きルールに関連するすべてのデータオブジェクトは、ガイド付きルールと同じプロジェクトパッケージに置く必要があります。同じパッケージに含まれるアセットはデフォルトでインポートされます。必要なデータオブジェクトとガイド付きルールを作成したら、ガイド付きルールデザイナーの **Data Objects** タブから、必要なデータオブジェクトがすべてリストされていることを検証したり、新規アイテム を追加してその他の既存データオブジェクトをインポートしたりできます。

2.2.5.2. 従業員のシフト数のバランスを取るガイド付きルールの作成

ガイド付きルール **BalanceEmployeesShiftNumber** は、可能な限りバランスを取るように従業員にシフトを割り当てるソフト制約を作成します。これは、シフトの分配が平等でなくなると増えるスコアペナルティーを作成することで行います。ルールによって実装されたスコア式により、Solver がよりバランスの取れるようにシフトを分散させます。



手順

1. Business Central で、**Menu → Design → Projects** に移動して、プロジェクト名をクリックします。

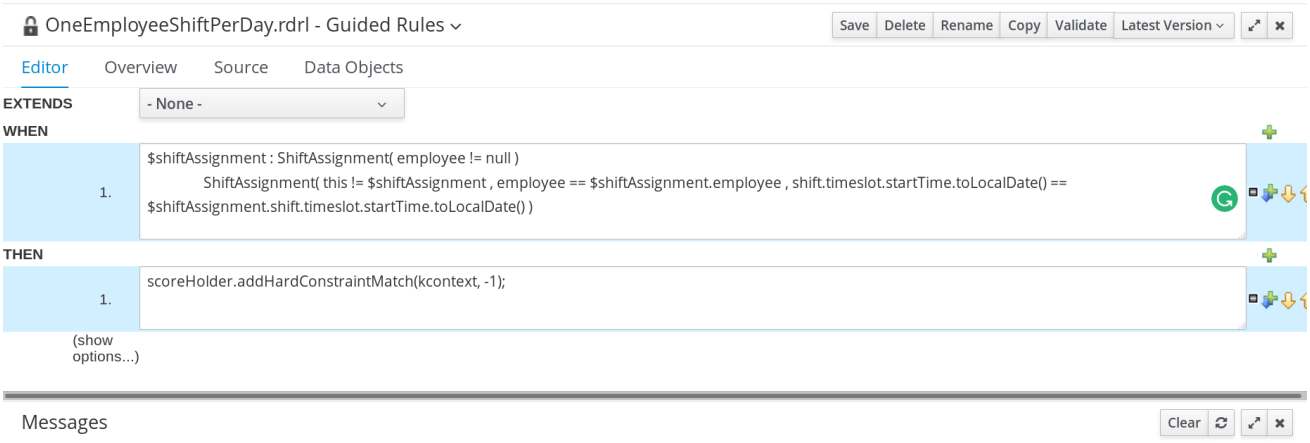
2. **Add Asset → Guided Rule** の順にクリックします。

3. Guided Rule 名 に **BalanceEmployeesShiftNumber** と入力し、Package で **employeerostering.employeerostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで ✚ をクリックして、**WHEN** 条件を追加します。
6. Add a condition to the ruleウィンドウで **Employee** を選択します。**+OK** をクリックします。
7. **Employee** 条件でクリックして制約を修正し、変数名 **$employee** を追加します。
8. WHEN 条件 **From Accumulate** を追加します。
    a. **From Accumulate** 条件の上で click to add patternをクリックし、ドロップダウンリストでファクトタイプ **Number** を選択します。
    b. 変数名 **$shiftCount** を **Number** 条件に追加します。
    c. **From Accumulate** 条件の下で click to add patternをクリックして、ドロップダウンリストで **ShiftAssignment** ファクトタイプを選択します。
    d. 変数名 **$shiftAssignment** を **ShiftAssignment** ファクトタイプに追加します。
    e. **ShiftAssignment** 条件を再度クリックし、Add a restriction on a field ドロップダウンリストで **employee** を選択します。
    f. **employee** 制約の横にあるドロップダウンリストで **equal to** を選択します。
    g. ドロップダウンボタンの横の  アイコンをクリックして変数を追加し、**Field value** ウィンドウで Bound variable をクリックします。
    h. ドロップダウンリストで **$employee** を選択します。
    i. Function ボックスに **count($shiftAssignment)** と入力します。
9. **THEN** フィールドで ✚ をクリックして**THEN** 条件を追加します。
10. **Add a new action** ウィンドウで Modify Soft Score を選択します。**+OK** をクリックします。
    a. ボックスに **-($shiftCount.intValue()*$shiftCount.intValue())** と入力します。
11. 右上隅の Validate をクリックし、ルール条件がすべて有効であることを確認しますルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
12. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は、『ガイド付きルールを使用したデシジョンサービスの作成』を参照してください。

2.2.5.3. 同じ日に複数のシフトを割り当てないようにするガイド付きルールの作成

ガイド付きルール **OneEmployeeShiftPerDay** は、同じ日の複数のシフトに従業員を割り当てないようにするハード制約を作成します。従業員の勤務表サンプルでは、この制約はガイド付きルールデザイナーを使用して作成されます。



手順

1. Business Central で、 **Menu → Design → Projects** に移動して、プロジェクト名をクリックします。
2. **Add Asset → Guided Rule** の順にクリックします。
3. Guided Rule 名に **OneEmployeeShiftPerDay** と入力し、Package で **employeerostering.employeerostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで をクリックして、**WHEN** 条件を追加します。
6. **Add a condition to the rule**ウィンドウから **Free form DRL** を選択します。
7. **Free form** の **DRL** ボックスに、以下の条件を入力します。

```
$shiftAssignment : ShiftAssignment( employee != null )
  ShiftAssignment( this != $shiftAssignment , employee == $shiftAssignment.employee
, shift.timeslot.startTime.toLocalDate() ==
$shiftAssignment.shift.timeslot.startTime.toLocalDate() )
```

この条件は、同じ日に別のシフトがすでに割り当てられている従業員にはシフトを割り当てることができないことを示しています。

8. **THEN** フィールドで をクリックして**THEN** 条件を追加します。
9. **Add a new action**ウィンドウから **Add Free form DRL**を選択します。
10. **Free form** の **DRL** ボックスに、以下の条件を入力します。

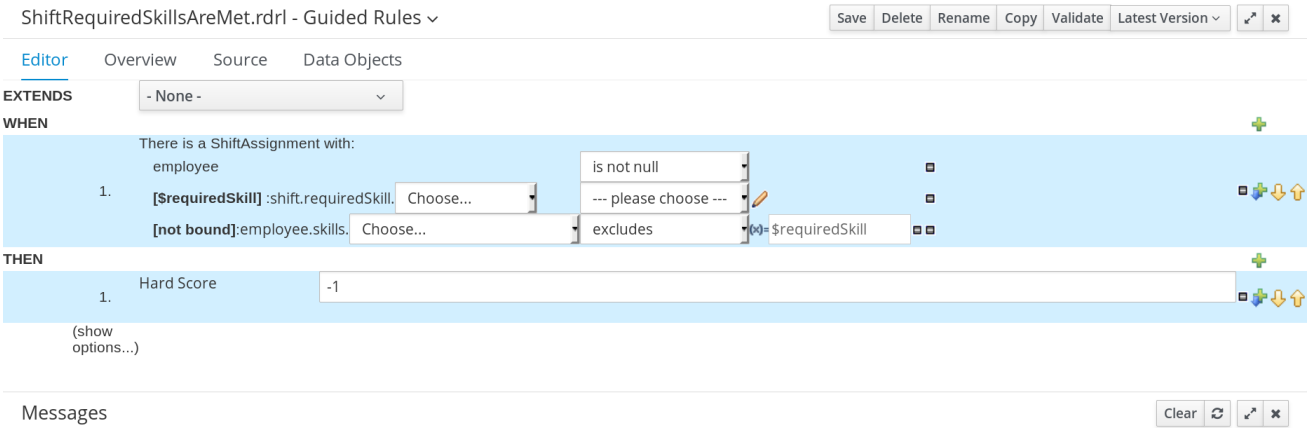
```
scoreHolder.addHardConstraintMatch(kcontext, -1);
```

11. 右上隅の **Validate** をクリックし、ルール条件がすべて有効であることを確認しますルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
12. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は、『ガイド付きルールを使用したデシジョンサービスの作成』を参照してください。

2.2.5.4. シフト要件にスキルを一致させるガイド付きルールの作成

ガイド付きルール **ShiftReqiredSkillsAreMet** は、すべてのシフトが、適切なスキルセットを持つ従業員に割り当てられるのを確認するハード制約を作成します。従業員の勤務表サンプルでは、この制約はガイド付きルールデザイナーを使用して作成されます。



手順

1. Business Central で、 **Menu → Design → Projects** に移動して、プロジェクト名をクリックします。
2. **Add Asset → Guided Rule** の順にクリックします。
3. Guided Rule 名に **ShiftReqiredSkillsAreMet** と入力し、Package で **employeerostering.employeerostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで をクリックして、**WHEN** 条件を追加します。
6. **Add a condition to the rule**ウィンドウで **ShiftAssignment** を選択します。**+OK** をクリックします。
7. **ShiftAssignment** 条件をクリックし、**Add a restriction on a field** ドロップダウンリストで **employee** を選択します。
8. デザイナーで、**employee** の横のドロップダウンリストをクリックし、**is not null** を選択します。
9. **ShiftAssignment** 条件をクリックし、**Expression editor** をクリックします。
    a. デザイナーで、**[not bound]** をクリックし、**Expression editor** を開き、式と変数 **$requiredSkill** をバインドします。**Set** をクリックします。
    b. デザイナーの **$requiredSkill** の横にあるドロップダウンリストで **shift** を選択し、その隣のドロップダウンリストで **requiredSkill** を選択します。
10. **ShiftAssignment** 条件をクリックし、**Expression editor** をクリックします。
    a. デザイナーで、**[not bound]** の横にあるドロップダウンリストで **employee** を選択し、その隣のドロップダウンリストで **skills** を選択します。
    b. その隣のドロップダウンリストでは **Choose** を選択したままにします。

c. その隣のドロップダウンボックスで、**please choose** を **excludes** に変更します。

d. **excludes** の横にある アイコンをクリックし、**Field value** ウィンドウで **New formula** ボタンをクリックします。

e. 式ボックスに **$requiredSkill** を追加します。

11. **THEN** フィールドで をクリックして**THEN** 条件を追加します。

12. **Add a new action** ウィンドウで **Modify Hard Score** を選択します。**+OK** をクリックします。

13. スコアアクションボックスに **-1** を入力します。

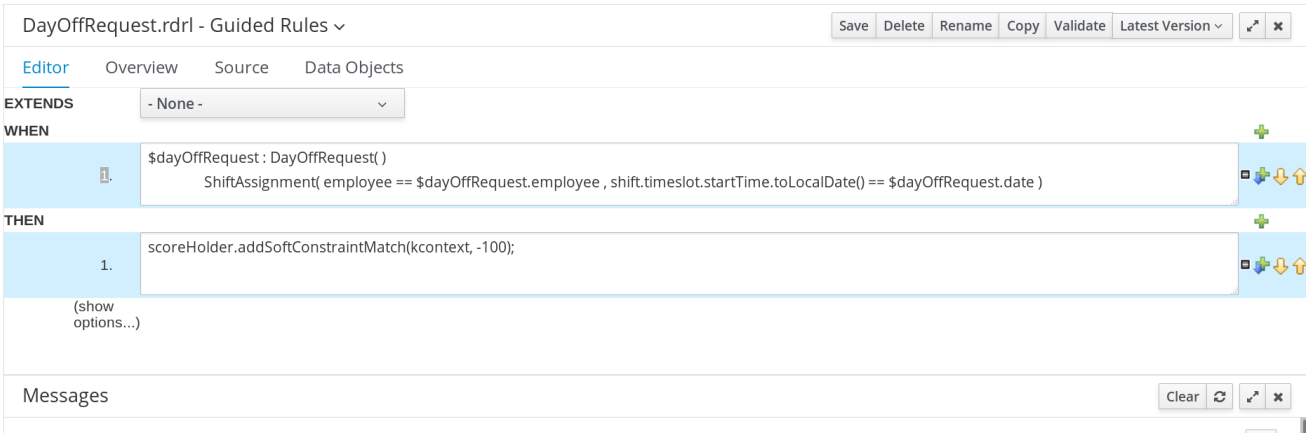
14. 右上隅の **Validate** をクリックし、ルール条件がすべて有効であることを確認しますルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。

15. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は、『ガイド付きルールを使用したデシジョンサービスの作成』を参照してください。

2.2.5.5. 休暇申請を管理するガイド付きルールの作成

ガイド付きルール **DayOffRequest** は、ソフト制約を作成します。この制約では、そのシフトに元々割り当てられていた従業員がその日に就業できなくなった場合に、別の従業員にシフトを再割り当てできるようにできます。従業員の勤務表サンプルでは、この制約はガイド付きルールデザイナーを使用して作成されます。



手順

1. **Business Central** で、**Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。

2. **Add Asset** → **Guided Rule** の順にクリックします。

3. **Guided Rule** 名に **DayOffRequest** と入力し、**Package** で **employeerostering.employeerostering** を選択します。

4. **OK** をクリックして、ルールアセットを作成します。

5. **WHEN** フィールドで をクリックして、**WHEN** 条件を追加します。

6. Add a condition to the ruleウィンドウから Free form DRL を選択します。
7. Free form の DRL ボックスに、以下の条件を入力します。

```
$dayOffRequest : DayOffRequest( )
  ShiftAssignment( employee == $dayOffRequest.employee ,
shift.timeslot.startTime.toLocalDate() == $dayOffRequest.date )
```

この条件は、休暇申請を行った従業員にシフトを割り当てている場合に、その従業員をその日のシフト割り当てから削除できることを示しています。

8. THEN フィールドで ✚ をクリックしてTHEN 条件を追加します。
9. Add a new action ウィンドウから Add Free form DRL を選択します。
10. Free form の DRL ボックスに、以下の条件を入力します。

```
scoreHolder.addSoftConstraintMatch(kcontext, -100);
```

11. 右上隅の Validate をクリックし、ルール条件がすべて有効であることを確認しますルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
12. Save をクリックして、ルールを保存します。

ガイド付きルールの作成方法は、『ガイド付きルールを使用したデシジョンサービスの作成』を参照してください。

2.2.6. 従業員の勤務表の Solver 設定の作成

Business Central に Solver 設定を作成して編集できます。Solver 設定デザイナーは、プロジェクトがデプロイされた後に実行できる Solver 設定を作成します。

前提条件

- Red Hat Process Automation Manager をダウンロードして、インストールしている。
- 従業員の勤務表サンプルに関連するアセットをすべて作成して設定している。

手順

1. Business Central で Menu → Projects をクリックし、使用するプロジェクトをクリックして開きます。
2. Assets パースペクティブで、Add Asset → Solver configuration をクリックします。
3. Create new Solver configuration ウィンドウで、Solver の名前 **EmployeeRosteringSolverConfig** と入力し、Ok をクリックします。
これにより、Solver configuration デザイナーが開きます。
4. Score Director Factory 設定セクションで、スコアリングルール定義を含む KIE ベースを定義します。従業員の勤務表サンプルプロジェクトは **defaultKieBase** を使用します。
    a. KIE ベースに定義した KIE セッションの中から 1 つ選択します。従業員の勤務表サンプルプロジェクトは **defaultKieSession** を使用します。

5. 右上の **Validate** をクリックし、**Score Director Factory** 設定が正しいことを確認します。妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、エラーが表示されなくなるまで妥当性確認を行います。

6. **Save** をクリックして、**Solver** 設定を保存します。

2.2.7. 従業員の勤務表プロジェクトに対する Solver の終了設定

一定期間が過ぎたら **Solver** が終了できるように設定できます。デフォルトでは、プランニングエンジンには、時間制限なく問題を解決できるように指定されています。

従業員の勤務表サンプルプロジェクトは、30 秒間実行するように設定されています。

前提条件

- 従業員の勤務表プロジェクトに、関連するすべてのアセットを作成し、「従業員の勤務表の Solver 設定の作成」 の説明に従って、**Business Central** に **Solver** 設定 **EmployeeRosteringSolverConfig** を作成している。

手順

1. **Assets** パースペクティブで **EmployeeRosteringSolverConfig** を開きます。これにより、**Solver** 設定 デザイナーが開きます。

2. **Termination** セクションで **Add** をクリックして、選択した論理グループに新しい終了要素を作成します。

3. ドロップダウンリストから、終了タイプ **Time spent** を選択します。これは、終了条件の入力フィールドとして追加されます。

4. 時間要素の横の矢印を使用して、経過時間を 30 秒に設定します。

5. 右上の **Validate** をクリックし、**Score Director Factory** 設定が正しいことを確認します。妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、エラーが表示されなくなるまで妥当性確認を行います。

6. **Save** をクリックして、**Solver** 設定を保存します。

2.3. REST API を使用した SOLVER へのアクセス

サンプルの **Solver** をデプロイするか再作成したら、**REST API** を使用してアクセスできます。

REST API を使用して **Solver** インスタンスを登録する必要があります。その後に、データセットを指定して、最適解を取得できます。

前提条件

- 本書の以前のセクションに従い、従業員の勤務表プロジェクトを設定し、デプロイしてください。「**Business Central** への従業員勤務表サンプルプロジェクトのデプロイメント」に記載のとおり、同じプロジェクトをデプロイするか、「従業員の勤務表サンプルプロジェクトの再作成」 に記載のとおり、プロジェクトを再作成できます。

2.3.1. REST API を使用した Solver の登録

Solver を使用するには、**REST API** を使用して **Solver** インスタンスを登録する必要があります。

各 Solver インスタンスで、一度に最適化できる計画問題の数は 1 つだけです。

手順

1. 以下のヘッダーを使用して HTTP 要求を作成します。

```
authorization: admin:admin
X-KIE-ContentType: xstream
content-type: application/xml
```

2. 以下の要求を使用して Solver を登録します。

PUT

http://localhost:8080/kie-server/services/rest/server/containers/employeerostering_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver

要求ボディー

```
<solver-instance>
  <solver-config-file>employeerostering/employeerostering/EmployeeRosteringSolverConfig.solver.xml</solver-config-file>
</solver-instance>
```

2.3.2. REST API を使用した Solver の呼び出し

Solver インスタンスを登録した後に、REST API を使用して、データセットを Solver に送信して、最適解を取得できます。

手順

1. 以下のヘッダーを使用して HTTP 要求を作成します。

```
authorization: admin:admin
X-KIE-ContentType: xstream
content-type: application/xml
```

2. 以下の例のように、データセットを含む Solver に要求を送信します。

POST

http://localhost:8080/kie-server/services/rest/server/containers/employeerostering_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver/state/solving

要求ボディー

```
<employeerostering.employeerostering.EmployeeRoster>
  <employeeList>
    <employeerostering.employeerostering.Employee>
      <name>John</name>
      <skills>
        <employeerostering.employeerostering.Skill>
          <name>reading</name>
```

```
        </employeerostering.employeerostering.Skill>
      </skills>
    </employeerostering.employeerostering.Employee>
    <employeerostering.employeerostering.Employee>
      <name>Mary</name>
      <skills>
        <employeerostering.employeerostering.Skill>
          <name>writing</name>
        </employeerostering.employeerostering.Skill>
      </skills>
    </employeerostering.employeerostering.Employee>
    <employeerostering.employeerostering.Employee>
      <name>Petr</name>
      <skills>
        <employeerostering.employeerostering.Skill>
          <name>speaking</name>
        </employeerostering.employeerostering.Skill>
      </skills>
    </employeerostering.employeerostering.Employee>
  </employeeList>
  <shiftList>
    <employeerostering.employeerostering.Shift>
      <timeslot>
        <startTime>2017-01-01T00:00:00</startTime>
        <endTime>2017-01-01T01:00:00</endTime>
      </timeslot>
      <requiredSkill
reference="../../../employeeList/employeerostering.employeerostering.Employee/skills/emplo
yeerostering.employeerostering.Skill"/>
    </employeerostering.employeerostering.Shift>
    <employeerostering.employeerostering.Shift>
      <timeslot reference="../../employeerostering.employeerostering.Shift/timeslot"/>
      <requiredSkill
reference="../../../employeeList/employeerostering.employeerostering.Employee[3]/skills/emp
loyeerostering.employeerostering.Skill"/>
    </employeerostering.employeerostering.Shift>
    <employeerostering.employeerostering.Shift>
      <timeslot reference="../../employeerostering.employeerostering.Shift/timeslot"/>
      <requiredSkill
reference="../../../employeeList/employeerostering.employeerostering.Employee[2]/skills/emp
loyeerostering.employeerostering.Skill"/>
    </employeerostering.employeerostering.Shift>
  </shiftList>
  <skillList>
    <employeerostering.employeerostering.Skill
reference="../../employeeList/employeerostering.employeerostering.Employee/skills/employe
erostering.employeerostering.Skill"/>
    <employeerostering.employeerostering.Skill
reference="../../employeeList/employeerostering.employeerostering.Employee[3]/skills/emplo
yeerostering.employeerostering.Skill"/>
    <employeerostering.employeerostering.Skill
reference="../../employeeList/employeerostering.employeerostering.Employee[2]/skills/emplo
yeerostering.employeerostering.Skill"/>
  </skillList>
  <timeslotList>
    <employeerostering.employeerostering.Timeslot
```

```
reference="../../shiftList/employeerostering.employeerostering.Shift/timeslot"/>
  </timeslotList>
  <dayOffRequestList/>
  <shiftAssignmentList>
    <employeerostering.employeerostering.ShiftAssignment>
      <shift reference="../../../shiftList/employeerostering.employeerostering.Shift"/>
    </employeerostering.employeerostering.ShiftAssignment>
    <employeerostering.employeerostering.ShiftAssignment>
      <shift reference="../../../shiftList/employeerostering.employeerostering.Shift[3]"/>
    </employeerostering.employeerostering.ShiftAssignment>
    <employeerostering.employeerostering.ShiftAssignment>
      <shift reference="../../../shiftList/employeerostering.employeerostering.Shift[2]"/>
    </employeerostering.employeerostering.ShiftAssignment>
  </shiftAssignmentList>
</employeerostering.employeerostering.EmployeeRoster>
```

3. 計画問題に最適解をリクエストします。

GET

http://localhost:8080/kie-server/services/rest/server/containers/employeerostering_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver/bestsolution

応答例

```
<solver-instance>
  <container-id>employee-rostering</container-id>
  <solver-id>solver1</solver-id>
  <solver-config-
file>employeerostering/employeerostering/EmployeeRosteringSolverConfig.solver.
xml</solver-config-file>
  <status>NOT_SOLVING</status>
  <score
scoreClass="org.optaplanner.core.api.score.buildin.hardsoft.HardSoftScore">0hard/0soft
</score>
  <best-solution class="employeerostering.employeerostering.EmployeeRoster">
    <employeeList>
      <employeerostering.employeerostering.Employee>
        <name>John</name>
        <skills>
          <employeerostering.employeerostering.Skill>
            <name>reading</name>
          </employeerostering.employeerostering.Skill>
        </skills>
      </employeerostering.employeerostering.Employee>
      <employeerostering.employeerostering.Employee>
        <name>Mary</name>
        <skills>
          <employeerostering.employeerostering.Skill>
            <name>writing</name>
          </employeerostering.employeerostering.Skill>
        </skills>
      </employeerostering.employeerostering.Employee>
      <employeerostering.employeerostering.Employee>
        <name>Petr</name>
```

```
        <skills>
          <employeerostering.employeerostering.Skill>
            <name>speaking</name>
          </employeerostering.employeerostering.Skill>
        </skills>
      </employeerostering.employeerostering.Employee>
    </employeeList>
    <shiftList>
      <employeerostering.employeerostering.Shift>
        <timeslot>
          <startTime>2017-01-01T00:00:00</startTime>
          <endTime>2017-01-01T01:00:00</endTime>
        </timeslot>
        <requiredSkill
reference="../../../employeeList/employeerostering.employeerostering.Employee/skills/emplo
yeerostering.employeerostering.Skill"/>
      </employeerostering.employeerostering.Shift>
      <employeerostering.employeerostering.Shift>
        <timeslot reference="../../employeerostering.employeerostering.Shift/timeslot"/>
        <requiredSkill
reference="../../../employeeList/employeerostering.employeerostering.Employee[3]/skills/emp
loyeerostering.employeerostering.Skill"/>
      </employeerostering.employeerostering.Shift>
      <employeerostering.employeerostering.Shift>
        <timeslot reference="../../employeerostering.employeerostering.Shift/timeslot"/>
        <requiredSkill
reference="../../../employeeList/employeerostering.employeerostering.Employee[2]/skills/emp
loyeerostering.employeerostering.Skill"/>
      </employeerostering.employeerostering.Shift>
    </shiftList>
    <skillList>
      <employeerostering.employeerostering.Skill
reference="../../employeeList/employeerostering.employeerostering.Employee/skills/employe
erostering.employeerostering.Skill"/>
      <employeerostering.employeerostering.Skill
reference="../../employeeList/employeerostering.employeerostering.Employee[3]/skills/emplo
yeerostering.employeerostering.Skill"/>
      <employeerostering.employeerostering.Skill
reference="../../employeeList/employeerostering.employeerostering.Employee[2]/skills/emplo
yeerostering.employeerostering.Skill"/>
    </skillList>
    <timeslotList>
      <employeerostering.employeerostering.Timeslot
reference="../../shiftList/employeerostering.employeerostering.Shift/timeslot"/>
    </timeslotList>
    <dayOffRequestList/>
    <shiftAssignmentList/>
    <score>0hard/0soft</score>
  </best-solution>
</solver-instance>
```

第3章 JAVA SOLVER のスタートガイド: クラウドバランシングの例

サンプルを使用して、Java コードを使用した基本的な Red Hat Business Optimizer Solver の開発を紹介します。

クラウドコンピューターを複数台所有し、そのクラウドコンピューターで複数のプロセスを実行する必要があると仮定します。コンピューターに各プロセスを割り当てる必要があります。

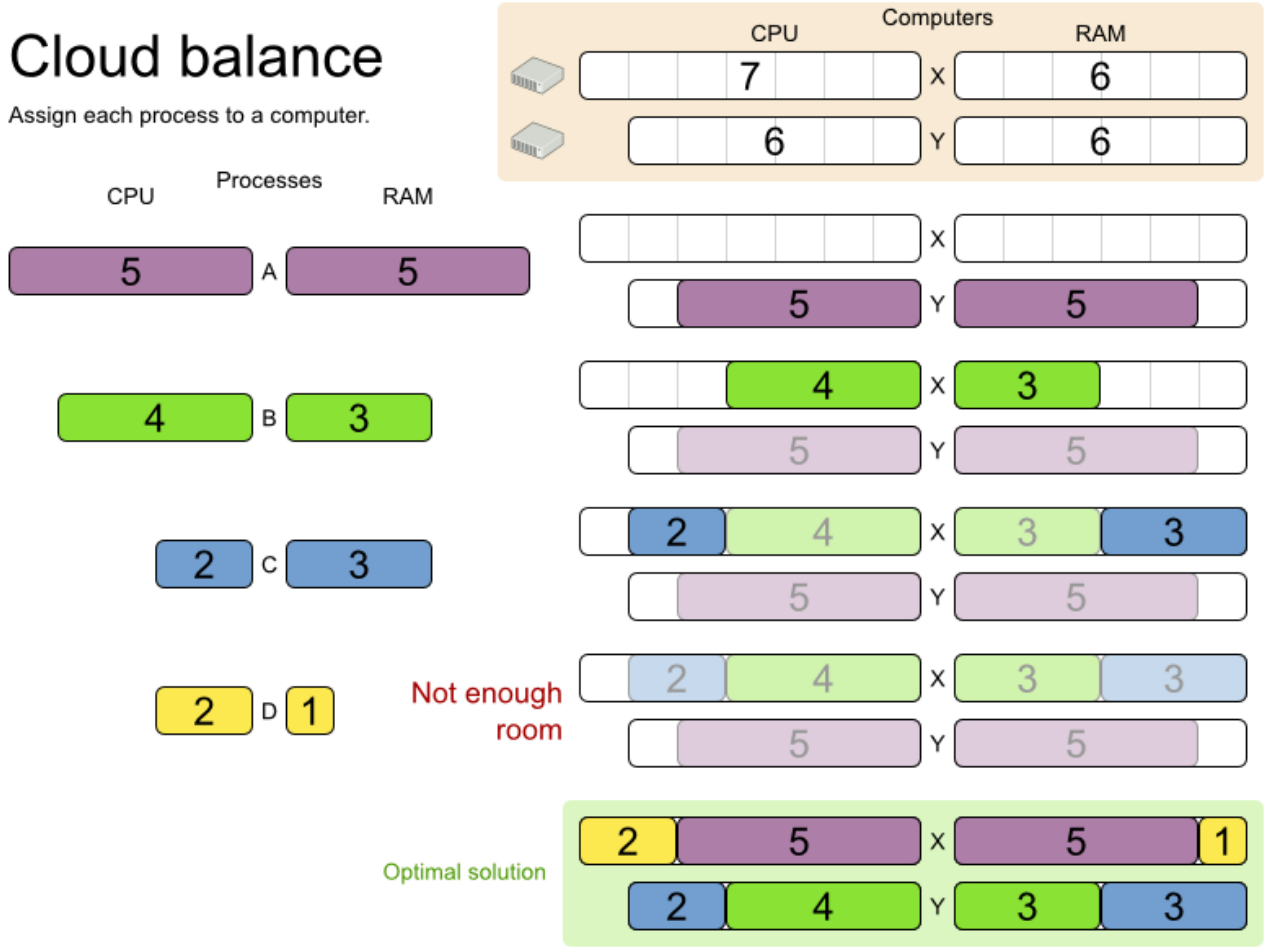
以下のハード制約を満たす必要があります。

- すべてのコンピューターで、プロセスの合計容量を処理するのに必要なハードウェア最小要件を満たす必要があります。
    - CPU 容量: コンピューターには、最低でも、そのコンピューターに割り当てられたプロセスで必要とされる合計の CPU 処理能力が必要です。
    - メモリー容量: コンピューターには、最低でも、そのコンピューターに割り当てられたプロセスで必要とされる合計のメモリーが必要です。
    - ネットワーク容量: コンピューターには、最低でも、そのコンピューターに割り当てられたプロセスで必要とされる合計のネットワーク帯域幅が必要です。

以下のソフト制約を最適化する必要があります。

- 1つまたは複数のプロセスが割り当てられたコンピューターにはそれぞれ、保守コストが発生します (コンピューターごとに固定)。
    - コスト: 合計保守コストを最小限に抑えます。

これは、ビンパッキング 問題にあたります。以下に、簡単なアルゴリズムを使用して、制約が 2 つ (CPU およびメモリー) あるコンピューター 2 台に、4 つのプロセスを割り当てるという簡単な例を紹介します。



ここで使用しているアルゴリズムは **FFD (First Fit Decreasing)** アルゴリズムです。このアルゴリズムでは、最初に大きいプロセスを割り当ててから、残りのスペースに小さいプロセスを割り当てていきます。図からも分かるように、これは黄色いプロセス **D** を割り当てる容量が残っていないため、最適ではありません。

Business Optimizer は、別の、より良いアルゴリズムを使用して、より適した解 (ソリューション) を見つけます。また、データ (プロセスやコンピューター) と制約 (ハードウェア要件などの制約) の両方を増やして評価します。

以下のまとめは、「マシンの再割当て (Google ROADEF 2012)」で説明されている、より多くの制約を使用した高度な実装や、この例に該当します。

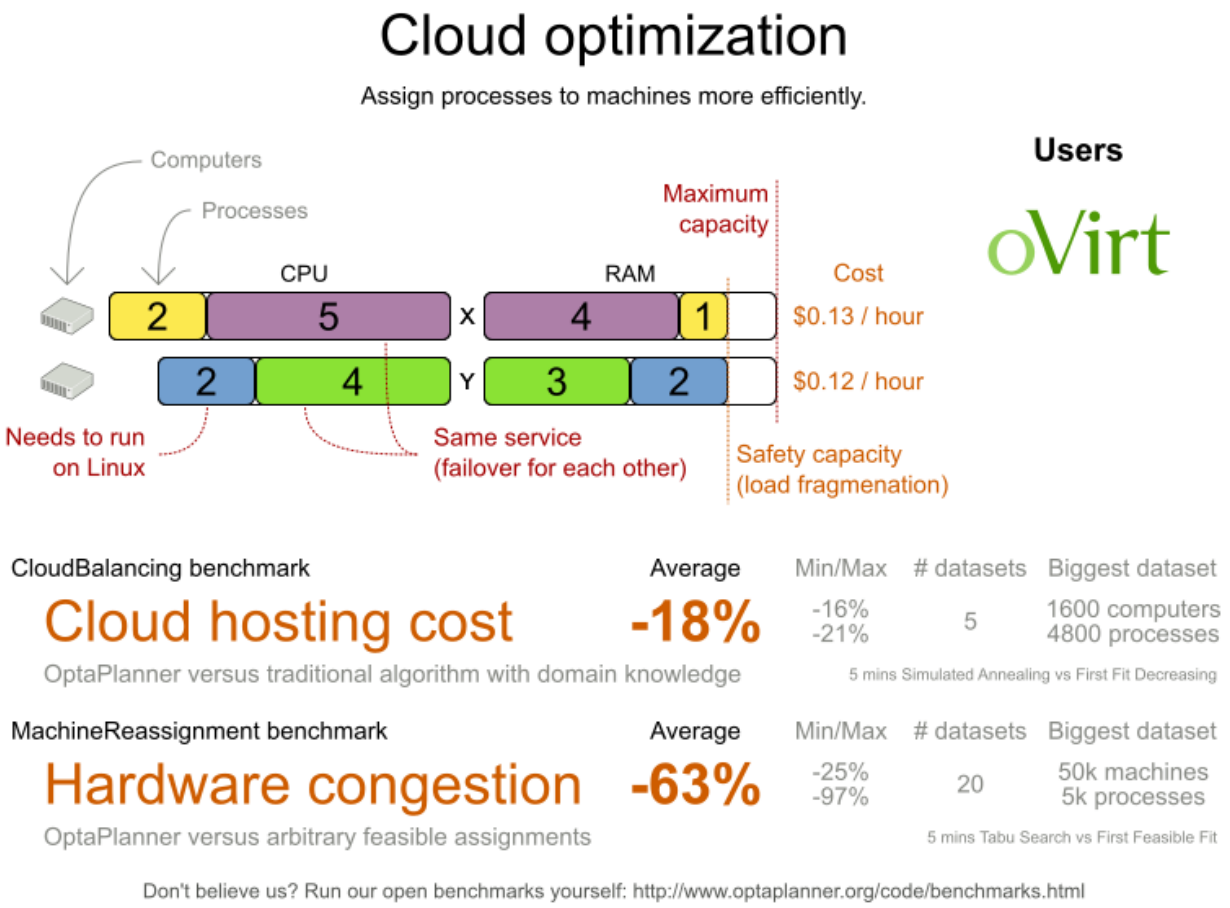


表3.1 クラウドのバランス問題の規模

問題の規模	コンピューター	プロセス	探索空間
コンピューター 2 台、プロセス 6 件	2	6	64
コンピューター 3 台、プロセス 9 件	3	9	10^4
コンピューター 4 台、プロセス 12 件	4	12	10^7
コンピューター 100 台、プロセス 300 件	100	300	10^600
コンピューター 200 台、プロセス 600 件	200	600	10^1380
コンピューター 400 台、プロセス 1200 件	400	1200	10^3122
コンピューター 800 台、プロセス 2400 件	800	2400	10^6967

3.1. ドメインモデルの設計

ドメインモデル を使用すると、どのクラスがプランニングエンティティーで、どのプロパティーがプランニング変数かが分かります。また、制約の簡素化、パフォーマンスの向上、これからのニーズに対する柔軟性の向上もサポートします。

3.1.1. ドメインモデルの設計

ドメインモデルを作成するには、問題の入力データを表現するオブジェクトをすべて定義します。この例では、オブジェクトはプロセスとコンピューターです。

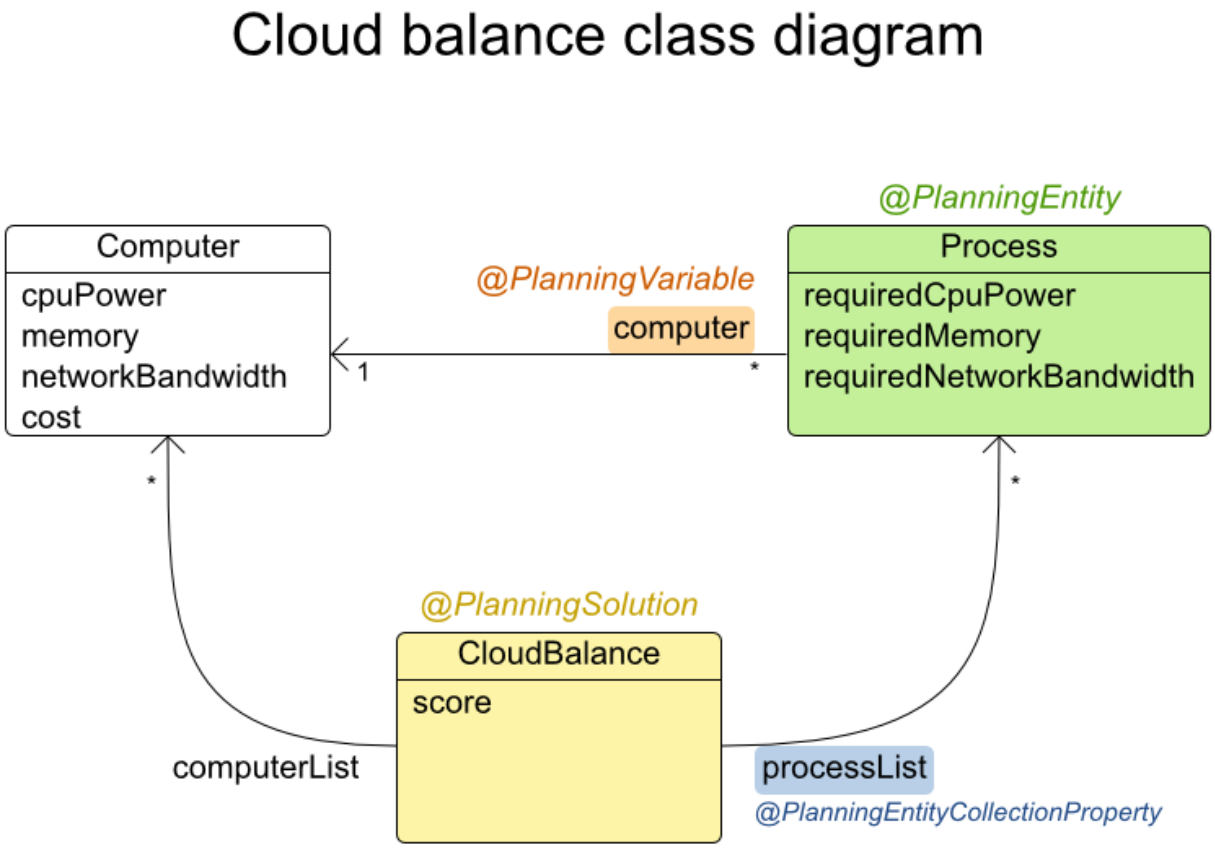
ドメインモデルの個別のオブジェクトは、ソリューションおよび入力データを含む、問題の完全なデータセットを表現する必要があります。以下の例では、このオブジェクトにコンピューターの一覧とプロセスの一覧を格納します。プロセスごとにコンピューターが 1 台割り当てられ、コンピューター間のプロセスの配分が解となります。

手順

1. ドメインモデルのクラス図を作成します。
2. 正規化して複製データを削除します。
3. クラスごとに サンプルインスタンス を記述します。サンプルインスタンスは、プランニングの目的に関連するエンティティープロパティーです。
   - **Computer**: 特定のハードウェアが搭載され、特定の保守コストが発生するコンピューターを表します。
     この例では、**cpuPower**、**memory**、**networkBandwidth**、および **cost** が **Computer** クラスのサンプルインスタンスです。
   - **Process**: デマンドのあるプロセスを表します。このプロセスは、Planner によって **Computer** に割り当てられます。
     Process のサンプルインスタンスは **requiredCpuPower**、**requiredMemory**、および **requiredNetworkBandwidth** です。
   - **CloudBalance**: コンピューター間のプロセスの配分を表します。CloudBalance には、特定のデータセットの **Computer** および **Process** がすべて含まれます。
     オブジェクトがすべてのデータセットおよび解を表す場合は、score を格納するサンプルインスタンスが必要です。Business Optimizer は異なる解のスコアを計算して、比較します。最高スコアの解が最適解となります。このため、**CloudBalance** のサンプルインスタンスは **score** になります。
4. プランニング中にどの関係 (またはフィールド) が変化するか判断します。
   - Planning entity (プランニングエンティティー): 解決中に Business Optimizer が変更可能なクラス。この例では、別のコンピューターにプロセスを移動できるため **Process** クラスです。
     - Business Optimizer が変更できない入力データを表現するクラスは、問題ファクト として知られています。
   - Planning variable (プランニング変数): 解決時に変化するプランニングエンティティークラスのプロパティー。この例では、**Process** クラスの **computer** プロパティーがそれにあたります。

- Planning solution (プランニングの解): 問題への解を表現するクラス。このクラスは、プランニングエンティティーをすべて含むデータセットを表す必要があります。この例では、**CloudBalance** クラスがそれにあたります。

以下の UML クラスの図では、Business Optimizer のコンセプトにすでにアノテーションが付けてあります。



examples/sources/src/main/java/org/optaplanner/examples/cloudbalancing/domain ディレクトリーに、この例のクラス定義が含まれています。

3.1.2. Computer クラス

Computer クラスは、データを保存する Java オブジェクトで、POJO (Plain Old Java Object) として知られています。通常、入力データを持つこの種のクラスが多くなります。

例3.1 CloudComputer.java

```
public class CloudComputer ... {

    private int cpuPower;
    private int memory;
    private int networkBandwidth;
    private int cost;

    ... // getters
}
```

3.1.3. Process クラス

Process クラスは、解決中に変更されるクラスです。

Business Optimizer に、**computer** プロパティーを変更できることを指示する必要があります。これには、クラスに **@PlanningEntity** のアノテーションを付けて、**getComputer()** ゲッターに **@PlanningVariable** のアノテーションを付けます。

当然ながら、Business Optimizer が解決中にプロパティーを変更できるように、この **computer** プロパティーにはセッターも必要です。

例3.2 CloudProcess.java

```
@PlanningEntity(...)
public class CloudProcess ... {

    private int requiredCpuPower;
    private int requiredMemory;
    private int requiredNetworkBandwidth;

    private CloudComputer computer;

    ... // getters

    @PlanningVariable(valueRangeProviderRefs = {"computerRange"})
    public CloudComputer getComputer() {
        return computer;
    }

    public void setComputer(CloudComputer computer) {
        computer = computer;
    }

    // ************************************************************************
    // Complex methods
    // ************************************************************************

    ...

}
```

Business Optimizer は、**computer** プロパティーに割り当てるのに、どの値を選択できるのかを把握しておく必要があります。これらの値は、プランニングの解の **CloudBalance.getComputerList()** メソッドから取得し、現在のデータセットに含まれる全コンピューターのリストを返します。

CloudProcess.getComputer() 上にある **@PlanningVariable** の **valueRangeProviderRefs** パラメーターは、**CloudBalance.getComputerList()** 上にある **@ValueRangeProvider** の **id** に一致する必要があります。

注記

ゲッターの代わりにフィールドにアノテーションも使用できます。

3.1.4. CloudBalance クラス

CloudBalance クラスには、 @PlanningSolution アノテーション。

このクラスは、すべてのコンピューターおよびプロセスの一覧を保持します。プランニングの問題と、(初期化されている場合は) プランニングの解の両方を表します。

CloudBalance クラスには、以下の主要な属性が含まれます。

- このクラスは、Business Optimizer が変更可能なプロセスコレクションを保持します。ゲッター **getProcessList()** に **@PlanningEntityCollectionProperty** アノテーションを付けて、Business Optimizer が変更できるプロセスを Business Optimizer で取得できるようにします。解を保存するには、Business Optimizer は、変更したプロセス一覧で、クラスの新規インスタンスを初期化します。

  1. これには **@PlanningScore** のアノテーションがついた **score** プロパティーも含まれており、このプロパティーは、現在の状態の解の **Score** を指します。Business Optimizer は、解のインスタンス向けに **Score** を計算すると自動的にこのプロパティーを更新するため、このプロパティーにはセッターが必要です。

  2. 特に、Drools でスコアの計算をする場合、**computerList** のプロパティーは **@ProblemFactCollectionProperty** のアノテーションを付けて、Business Optimizer がコンピューターのリスト (問題ファクト) を取得し、デシジョンエンジンに公開できるようにする必要があります。

例3.3 CloudBalance.java

```
@PlanningSolution
public class CloudBalance ... {

    private List<CloudComputer> computerList;

    private List<CloudProcess> processList;

    private HardSoftScore score;

    @ValueRangeProvider(id = "computerRange")
    @ProblemFactCollectionProperty
    public List<CloudComputer> getComputerList() {
        return computerList;
    }

    @PlanningEntityCollectionProperty
    public List<CloudProcess> getProcessList() {
        return processList;
    }

    @PlanningScore
    public HardSoftScore getScore() {
        return score;
    }

    public void setScore(HardSoftScore score) {
        this.score = score;
    }
```

```
    ...
}
```

3.2. クラウドバランシングの HELLO WORLD の実行

「hello world」アプリケーションサンプルを実行して、Solver を例示します。

手順

1. お好きな IDE にこの例をダウンロードして設定します。IDE へのサンプルのダウンロードおよび設定方法は、「IDE (IntelliJ、Eclipse、または Netbeans) での Red Hat Business Optimizer サンプルの実行」 を参照してください。

2. 以下の主要クラス **org.optaplanner.examples.cloudbalancing.app.CloudBalancingHelloWorld** で実行設定を作成します。
デフォルトでは、クラウドバランシングの Hello World は 120 秒間実行するように設定されています。

結果

このアプリケーションは、以下のコードを実行します。

例3.4 CloudBalancingHelloWorld.java

```
public class CloudBalancingHelloWorld {

    public static void main(String[] args) {
        // Build the Solver
        SolverFactory<CloudBalance> solverFactory =
SolverFactory.createFromXmlResource("org/optaplanner/examples/cloudbalancing/solver/
cloudBalancingSolverConfig.xml");
        Solver<CloudBalance> solver = solverFactory.buildSolver();

        // Load a problem with 400 computers and 1200 processes
        CloudBalance unsolvedCloudBalance = new
CloudBalancingGenerator().createCloudBalance(400, 1200);

        // Solve the problem
        CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);

        // Display the result
        System.out.println("\nSolved cloudBalance with 400 computers and 1200
processes:\n" + toDisplayString(solvedCloudBalance));
    }

    ...
}
```

このコードサンプルにより、以下が行われます。

1. Solver の設定をもとに **Solver** を構築します (ここでは、クラスパスの XML ファイル **cloudBalancingSolverConfig.xml** を使用します)。
Solver の構築がこの手順で最も複雑な部分です。詳細は、「Solver の設定」 を参照してください。

```
        SolverFactory<CloudBalance> solverFactory =
SolverFactory.createFromXmlResource(

"org/optaplanner/examples/cloudbalancing/solver/cloudBalancingSolverConfig.xml");
        Solver solver<CloudBalance> = solverFactory.buildSolver();
```

2. 問題を読み込みます。
CloudBalancingGenerator が無作為に問題を生成します。これは、たとえば、データベースなどから、実際の問題を読み込むクラスに置き換えてください。

```
        CloudBalance unsolvedCloudBalance = new
CloudBalancingGenerator().createCloudBalance(400, 1200);
```

3. 問題を解決します。

```
        CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);
```

4. 結果を表示します。

```
        System.out.println("\nSolved cloudBalance with 400 computers and 1200
processes:\n"
                + toDisplayString(solvedCloudBalance));
```

3.3. SOLVER の設定

Solver の設定ファイルは、解決のプロセスがどのように機能するかを指定します。このファイルはコードの一部とみなされます。このファイルの名前は、**examples/sources/src/main/resources/org/optaplanner/examples/cloudbalancing/solver/cloudBalancingSolverConfig.xml** になります。

例3.5 cloudBalancingSolverConfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<solver>
  <!-- Domain model configuration -->
  <scanAnnotatedClasses/>

  <!-- Score configuration -->
  <scoreDirectorFactory>

<easySoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.Cloud
BalancingEasyScoreCalculator</easyScoreCalculatorClass>
    <!--
<scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRules.drl
</scoreDrl>-->
  </scoreDirectorFactory>

  <!-- Optimization algorithms configuration -->
```

```
  <termination>
    <secondsSpentLimit>30</secondsSpentLimit>
  </termination>
</solver>
```

Solver の設定は、3 つの部分で構成されます。

1. ドメインモデル設定: Business Optimizer が変更可能なものは何ですか?
   Business Optimizer にドメインクラスを指定する必要があります。ここでは、(**@PlanningEntity** または **@PlanningSolution** アノテーションに対して) クラスパス内の全クラスを自動的にスキャンします。

   ```
   <scanAnnotatedClasses/>
   ```

2. スコアの設定: Business Optimizer はどのようにプランニング変数を最適化しますか? 目的は何ですか?
   ここでは、ハード制約とソフト制約を使用するため、**HardSoftScore** を使用します。ただし、Business Optimizer に、ビジネス要件に合ったスコアの計算方法を指定する必要があります。後ほど、スコアの計算方法を 2 種類 (基本的な Java 実装の使用、または Drools DRL の使用) 紹介します。

   ```
   <scoreDirectorFactory>

   <easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancingEasyScoreCalculator</easyScoreCalculatorClass>
     <!--
   <scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRules.drl</scoreDrl>-->
    </scoreDirectorFactory>
   ```

3. 最適化アルゴリズムの設定: Business Optimizer をどのように最適化しますか? この例では、(最適化アルゴリズムが明示的に設定されていないため) 30 秒間、デフォルトの最適化アルゴリズムを使用します。

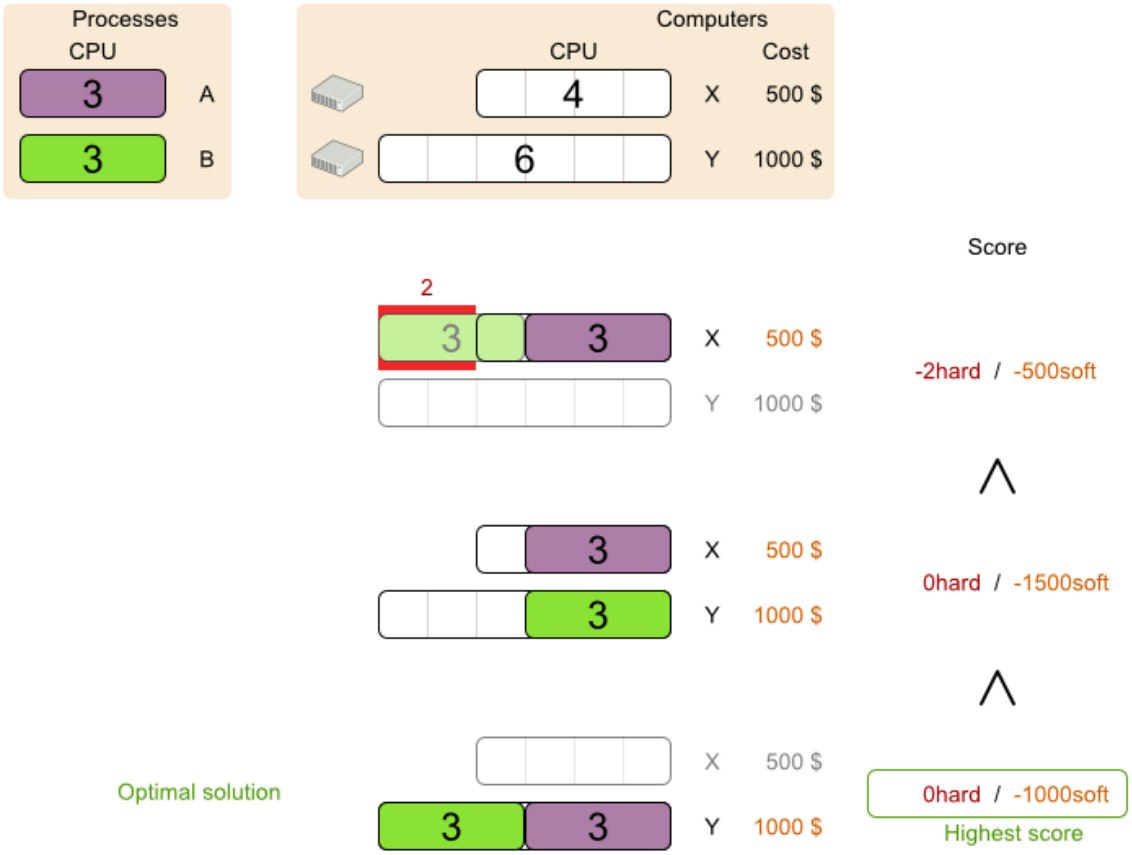
   ```
   <termination>
     <secondsSpentLimit>30</secondsSpentLimit>
   </termination>
   ```

   Business Optimizer は、数秒 (リアルタイム計画機能を使用する場合は 15 ミリ秒未満になる場合も) でも良い結果は得られるはずですが、時間が長くなればなるほど、結果は良くなります。高度なユースケースでは、ハードな時間制限以外に、終了の条件を使用することが適しています。

   デフォルトのアルゴリズムでも、人間の計画担当者やほとんどの社内実装を簡単に上回っています。高度なベンチマーク機能を使用して微調整を強化し、さらに良い結果を得ることができます。

3.4. スコアの設定

Business Oprtimizer は、スコア が最も高い ソリューション を探します。この例では、**HardSoftScore** を使用し、Business Oprtimizer がハード制約に違反していない (またはハードウェア要件を満たす) ソリューションと、ソフト制約に違反する数が最も少ない (メンテナンスコストが最も低い) ソリューションを探します。



当然ながら、Business Optimizer には、これらのドメイン固有のスコア制約についても指定する必要があります。Java 言語または Drools 言語を使用して制約を定義できます。

3.4.1. Java を使用したスコア計算の設定

スコア関数を定義する方法の 1 つに、Plain Java での **EasyScoreCalculator** インターフェース実装があります。

手順

1. **cloudBalancingSolverConfig.xml** ファイルで設定を追加するか、アンコメントします。

```
  <scoreDirectorFactory>

<easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.C
loudBalancingEasyScoreCalculator</easyScoreCalculatorClass>
  </scoreDirectorFactory>
```

2. **calculateScore(Solution)** メソッドを実装して **HardSoftScore** インスタンスを返します。

例3.6 CloudBalancingEasyScoreCalculator.java

```
public class CloudBalancingEasyScoreCalculator implements
EasyScoreCalculator<CloudBalance> {

  /**
   * A very simple implementation. The double loop can easily be removed by
```

```
    using Maps as shown in
     * {@link
CloudBalancingMapBasedEasyScoreCalculator#calculateScore(CloudBalance)}.
     */
    public HardSoftScore calculateScore(CloudBalance cloudBalance) {
        int hardScore = 0;
        int softScore = 0;
        for (CloudComputer computer : cloudBalance.getComputerList()) {
            int cpuPowerUsage = 0;
            int memoryUsage = 0;
            int networkBandwidthUsage = 0;
            boolean used = false;

            // Calculate usage
            for (CloudProcess process : cloudBalance.getProcessList()) {
                if (computer.equals(process.getComputer())) {
                    cpuPowerUsage += process.getRequiredCpuPower();
                    memoryUsage += process.getRequiredMemory();
                    networkBandwidthUsage += process.getRequiredNetworkBandwidth();
                    used = true;
                }
            }

            // Hard constraints
            int cpuPowerAvailable = computer.getCpuPower() - cpuPowerUsage;
            if (cpuPowerAvailable < 0) {
                hardScore += cpuPowerAvailable;
            }
            int memoryAvailable = computer.getMemory() - memoryUsage;
            if (memoryAvailable < 0) {
                hardScore += memoryAvailable;
            }
            int networkBandwidthAvailable = computer.getNetworkBandwidth() -
networkBandwidthUsage;
            if (networkBandwidthAvailable < 0) {
                hardScore += networkBandwidthAvailable;
            }

            // Soft constraints
            if (used) {
                softScore -= computer.getCost();
            }
        }
        return HardSoftScore.valueOf(hardScore, softScore);
    }

}
```

上記のコードを最適化し、**Map** を使用して **processList** を 1 回だけ反復した場合でも、インクリメンタルスコアの計算が行われないため、処理が遅いままです。

これを修正するには、インクリメント Java スコア計算か、Drools スコア計算を使用します。インクリメント Java スコア計算については、本書では触れません。

3.4.2. Drools を使用したスコア計算の設定

Drools ルール言語 (DRL) を使用して制約を定義できます。Drools スコア計算はインクリメント計算を使用します。この計算では、1 つまたは複数のスコアルールとしてすべてのスコア制約が記述されます。

スコア計算のデシジョンエンジンを使用すると、デシジョンテーブル (XLS または Web ベース)、Business Central をはじめとしたサポート対象の機能など、Drools の他の技術と統合できます。

手順

1. クラスパスに **scoreDrl** リソースを追加してスコア機能としてデシジョンエンジンを使用します。**cloudBalancingSolverConfig.xml** ファイルで設定を追加するか、アンコメントします。

```
  <scoreDirectorFactory>

<scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRule
s.drl</scoreDrl>
  </scoreDirectorFactory>
```

2. ハード制約を作成します。これらの制約で、すべてのコンピューターに、十分な CPU、メモリー、ネットワーク帯域幅が割り当てられ、全プロセスがサポートされるようになります。

例3.7 cloudBalancingScoreRules.drl - ハード制約

```
...

import org.optaplanner.examples.cloudbalancing.domain.CloudBalance;
import org.optaplanner.examples.cloudbalancing.domain.CloudComputer;
import org.optaplanner.examples.cloudbalancing.domain.CloudProcess;

global HardSoftScoreHolder scoreHolder;

//
################################################################################
####
// Hard constraints
//
################################################################################
####

rule "requiredCpuPowerTotal"
    when
        $computer : CloudComputer($cpuPower : cpuPower)
        accumulate(
            CloudProcess(
                computer == $computer,
                $requiredCpuPower : requiredCpuPower);
            $requiredCpuPowerTotal : sum($requiredCpuPower);
            $requiredCpuPowerTotal > $cpuPower
        )
    then
        scoreHolder.addHardConstraintMatch(kcontext, $cpuPower -
$requiredCpuPowerTotal);
end
```

```
rule "requiredMemoryTotal"
    ...
end

rule "requiredNetworkBandwidthTotal"
    ...
end
```

3. ソフト制約を作成します。この制約は、保守コストを最小限に抑えます。ハード制約に該当する場合にのみ適用されます。

例3.8 **cloudBalancingScoreRules.drl** - ソフト制約

```
//
################################################################################
####
// Soft constraints
//
################################################################################
####

rule "computerCost"
    when
        $computer : CloudComputer($cost : cost)
        exists CloudProcess(computer == $computer)
    then
        scoreHolder.addSoftConstraintMatch(kcontext, - $cost);
end
```

3.5. SOLVER の他の開発

上記の例が機能するようになったので、さらに開発を進めてみてください。たとえば、ドメインモデルを改良して、以下のような制約を追加してみてください。

- すべての プロセス が サービス に属する。コンピューターはクラッシュする可能性があるため、同じサービスを実行するプロセスは、別のコンピューターに割り当てる必要がある (または割り当てなければならない)。
- すべての コンピューター が ビル に設置されている。ビルが火災にあう可能性があるため、同じサービスのプロセスは、別のビルに設置されているコンピューターに割り当てる必要がある (割り当てなければならない)。

第4章 RED HAT BUSINESS OPTIMIZER で提供される例

Red Hat Process Automation Manager には、Red Hat Business Optimizer のサンプルが複数同梱されています。たとえばコードなどを確認して、ニーズに合ったものに変更できます。

注記

Red Hat は、Red Hat Process Automation Manager ディストリビューションに含まれるコードサンプルのサポートはしていません。

4.1. サンプルのダウンロードおよび実行

Red Hat Software Downloads の Web サイトから Red Hat Business Optimizer のサンプルをダウンロードして、実行できます。

4.1.1. Red Hat Business Optimizer のサンプルのダウンロード

Red Hat Process Automation Manager のアドオンパッケージの一部として例をダウンロードできます。

手順

1. Software Downloads ページから **rhpam-7.7.0-add-ons.zip** ファイルをダウンロードします。
2. ファイルを展開します。
3. **rhpam-7.7-planner-engine.zip** ファイルを展開したディレクトリーから展開します。

結果

展開した **rhpam-7.7-planner-engine** ディレクトリーの以下のサブディレクトリーに、サンプルのソースコードが含まれています。*** examples/sources/src/main/java/org/optaplanner/examples * examples/sources/src/main/resources/org/optaplanner/examples * webexamples/sources/src/main/java/org/optaplanner/examples * webexamples/sources/src/main/resources/org/optaplanner/examples**

「Business Optimizer サンプルの表」には、各サンプルで使用するディレクトリーの名前を記載しています。

4.1.2. Business Optimizer サンプルの実行

Red Hat Business Optimizer には、さまざまなユースケースのデモとして多数のサンプルが含まれています。

前提条件

- 例をダウンロードして展開しました。これらのアクションに関する詳細は、「Red Hat Business Optimizer のサンプルのダウンロード」を参照してください。

手順

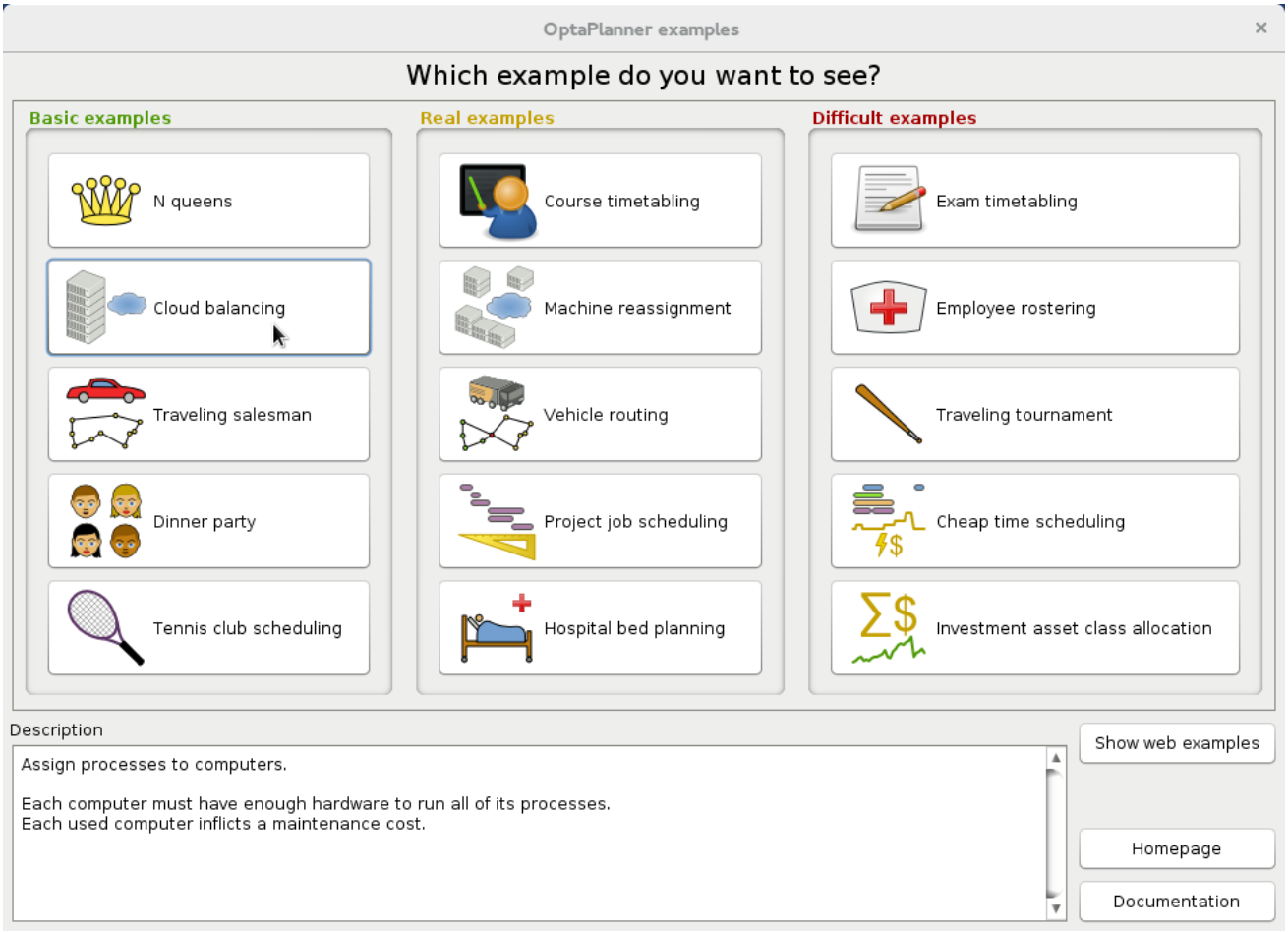
1. **rhpam-7.7.0-planner-engine** フォルダーで、**examples** ディレクトリーを開き、適切なスクリプトを使用してサンプルを実行します。
   Linux または Mac の場合:

```
$ cd examples
$ ./runExamples.sh
```

Windows:

```
$ cd examples
$ runExamples.bat
```

GUI アプリケーションウィンドウからサンプルを選択して実行します。



注記

Red Hat Business Optimizer 自体は GUI に依存していません。デスクトップと同じように、サーバーまたはモバイル JVM 上でも実行できます。

4.1.3. IDE (IntelliJ、Eclipse、または Netbeans) での Red Hat Business Optimizer サンプルの実行

IntelliJ、Eclipse、Netbeans など統合開発環境 (IDE) を使用する場合は、お使いの開発環境にダウンロードした Red Hat Business Optimizer の例を実行できます。

前提条件

- 例をダウンロードして展開しました。これらのアクションに関する詳細は、「Red Hat Business Optimizer のサンプルのダウンロード」 を参照してください。

手順

1. 新規プロジェクトとして Red Hat Business Optimizer サンプルを開きます。

   a. IntelliJ または Netbeans の場合は、新規プロジェクトとして **examples/sources/pom.xml** を開きます。Maven 統合の指示に従い、インストールを進めてください。この手順では、残りのステップは省略します。

   b. Eclipse の場合は、**examples/sources** ディレクトリーで新規プロジェクトを開きます。

2. **binaries** ディレクトリーおよび **examples/binaries** ディレクトリーにあるすべての JAR を、クラスパスに追加します (**examples/binaries/optaplanner-examples-*.jar** ファイルを除く)。

3. Java のソースディレクトリー **src/main/java** と Java のリソースディレクトリー **src/main/resources** を追加します。

4. 実行設定を作成します。

   - メインクラス: **org.optaplanner.examples.app.OptaPlannerExamplesApp**
   - VM パラメーター (任意): **-Xmx512M -server -Dorg.optaplanner.examples.dataDir=examples/sources/data**
   - 作業ディレクトリー: **examples/sources**

5. 実行設定を実行します。

4.1.4. Web のサンプルの実行

GUI のサンプル以外に、Red Hat Process Automation Manager には、Red Hat Business Optimizer 用の一連の Web サンプルが含まれます。Web の例を以下に示します。

- 配送経路: Leaflet または Google Maps の視覚化機能どちらかを使用して、さまざまな顧客が必要とするアイテムをすべて回収する最短経路を計算します。
- クラウドバランシング: 異なる仕様およびコストのコンピューターにプロセスを割り当てます。

前提条件

- Red Hat Process Automation Manager アドオンパッケージから Red Hat Business Optimizer のサンプルをダウンロードして、展開しました。手順については、「Red Hat Business Optimizer のサンプルのダウンロード」 を参照してください。

Web のサンプルは、以下の API など、複数の JEE API を実行する必要があります。

- Servlet
- JAX-RS
- CDI

これらの API は、Bussiness Optimizer 自体には必要ありません。

手順

1. JBoss EAP、WildFly などの JEE アプリケーションサーバーをダウンロードして展開します。

2. 展開した **rhpam-7.7.0-planner-engine** ディレクトリーで、**webexamples/binaries** サブディレクトリーを開き、JEE アプリケーションサーバーに **optaplanner-webexamples-*.war** ファイルをデプロイします。
スタンドアロンモードで JBoss EAP を使用する場合は、**optaplanner-webexamples-*.war** ファイルを **JBOSS_home/standalone/deployments** ディレクトリーに追加して実行できます。

3. Web ブラウザーで http://localhost:8080/optaplanner-webexamples/ のアドレスを開きます。

4.2. BUSINESS OPTIMIZER サンプルの表

Business Optimizer サンプルには、教育関連のコンテストで出題された問題を解決するものもあります。以下の表の **Contest** 列には、このようなコンテストが掲載されています。また、コンテストの目的として、現実的 か、非現実的 かの識別をしています。現実的なコンテスト とは、独立した公式コンテストを指します。

現実的なコンテスト とは、以下の基準を満たす、独立した公式コンテストを指します。

- 明確に定義された実際のユースケースであること
- 実際に制約があること
- 実際のデータセットが複数あること
- 特定のハードウェアで特定の時間内に結果を再現できること
- 教育機関および/または企業の運用研究コミュニティーが真剣に参加していること

現実的なコンテストでは、競合のソフトウェアや教育研究と Business Optimizer を客観的に比較できます。

表4.1 サンプルの概要

例ドメイン	サイズ	ディレクトリー名 コンテスト
Nエンティティークラス1つ クイ(変数1つ) ーン	エンティティー ⇐ **256** 値 ⇐ **256** 探索空間 ⇐ **10^616**	無**nqueens** 意 味 （ 不 正 が 可 能 ）

例ドメイン	サイズ	ディレクトリー名 コンテスト
クラウドバランシング	エンティティークラス1つ (変数1つ) エンティティー ⇐ **2400** 値 ⇐ **800** 探索空間 ⇐ **10^6967**	**cloudbalancing** いいえ(弊社が定義)
巡回セールスマン	エンティティークラス1つ (連鎖変数1つ) エンティティー ⇐ **980** 値 ⇐ **980** 探索空間 ⇐ **10^2504**	**tsp** 現実的でないTSPWeb
ディナーパーティー	エンティティークラス1つ (変数1つ) エンティティー ⇐ **144** 値 ⇐ **72** 探索空間 ⇐ **10^310**	**dinnerParty** 現実的でない
テニスクラブのスケジュール	エンティティークラス1つ (変数1つ) エンティティー ⇐ **72** 値 ⇐ **7** 探索空間 ⇐ **10^60**	**tennis** いいえ(弊社が定義)

例ドメイン	サイズ	ディレクトリー名コンテスト
会議のスケジュール	エンティティークラス 1 つ (変数 2 つ)	エンティティー ⇐ **10** 値 ⇐ **320** および ⇐**5** 探索空間 ⇐ **10^320**	**meetingscheduling** いいえ (弊社が定義)
コースの時間割	エンティティークラス 1 つ (変数 2 つ)	エンティティー ⇐ **434** 値 ⇐ **25** および ⇐**20** 探索空間 ⇐ **10^1171**	**curriculumCourse** 現実的 ITC2007track3
マシンの再割当て	エンティティークラス 1 つ (変数 1 つ)	エンティティー ⇐ **50000** 値 ⇐ **5000** 探索空間 ⇐ **10^184948**	**machineReassignment** ほぼ現実的 ROADEF2012

例	ドメイン	サイズ	ディレクトリー名
配送経路	エンティティークラス1つ (連鎖変数1つ) シャドウエンティティークラス1つ (自動シャドウ変数1つ)	エンティティー ⇐ **2740** 値 ⇐ **2795** 探索空間 ⇐ **10^8380**	**vehiclerouting** 現実的でないVRPWeb
時間枠がある中での配送経路	配送経路すべて (シャドウ変数1つ)	エンティティー ⇐ **2740** 値 ⇐ **2795** 探索空間 ⇐ **10^8380**	**vehiclerouting** 現実的でないVRPWeb
プロジェクトジョブのスケジュール	エンティティークラス1つ (変数2つ) (シャドウ変数1つ)	エンティティー ⇐ **640** 値 ⇐ **?** および ⇐ **?** 探索空間 ⇐ **?**	**projectjobscheduling** ほぼ現実的MISTA2013

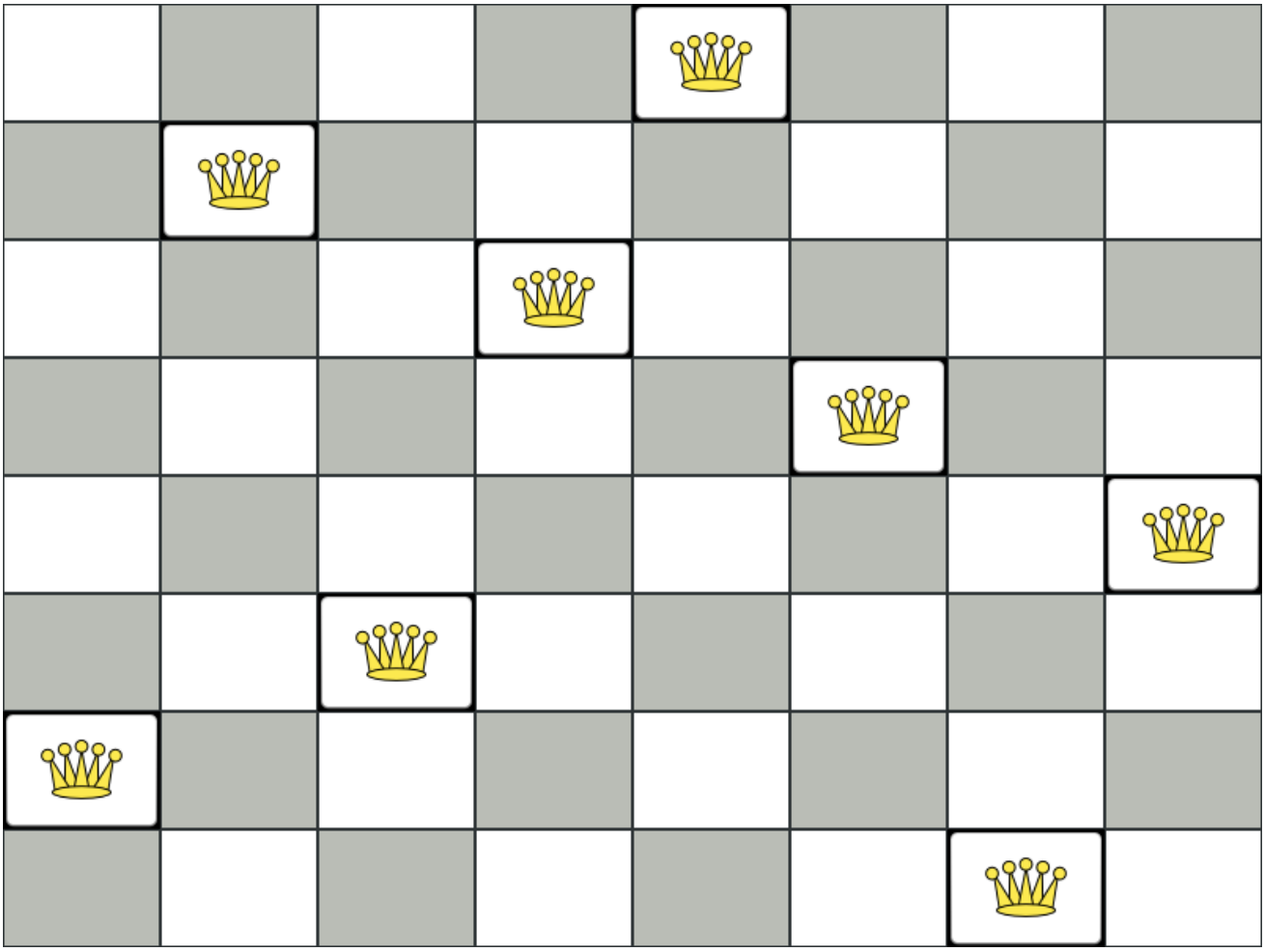
例	ドメイン	サイズ	ディレクトリー名	コンテスト
タスクの割り当て	エンティティークラス 1 つ (連鎖変数 1 つ) シャドウエンティティークラス 1 つ (シャドウ変数 1 つ) (自動シャドウ変数 1 つ)	エンティティー ⇐ **500** 値 ⇐ **520** 探索空間 ⇐ **10^1168**	**taskassigning**	いいえ (弊社が定義)
試験の時間割	エンティティークラス 2 つ (同じ階層) (変数 2 つ)	エンティティー ⇐ **1096** 値 ⇐ **80** および ⇐**49** 探索空間 ⇐ **10^3374**	**examination**	現実的 ITC2007track1
看護師の勤務表	エンティティークラス 1 つ (変数 1 つ)	エンティティー ⇐ **752** 値 ⇐ **50** 探索空間 ⇐ **10^1277**	**nurserostering**	現実的 INRC2010

例	ドメイン	サイズ	ディレクトリー名	コンテスト
巡回トーナメント	エンティティークラス 1 つ (変数 1 つ)	エンティティー ⇐ **1560** 値 ⇐ **78** 探索空間 ⇐ **10^2301**	**travelingtournament**	現実的でない TTP
コストを抑えるスケジュール	エンティティークラス 1 つ (変数 2 つ)	エンティティー ⇐ **500** 値 ⇐ **100** および ⇐**288** 探索空間 ⇐ **10^20078**	**cheaptimescheduling**	ほぼ現実的 ICON Energy
投資	エンティティークラス 1 つ (変数 1 つ)	エンティティー ⇐ **11** 値 = **1000** 探索空間 ⇐ **10^4**	**investment**	いいえ (弊社が定義)
会議スケジュール	エンティティークラス 1 つ (変数 2 つ)	エンティティー ⇐ **216** 値 ⇐ **18** および ⇐**20** 探索空間 ⇐ **10^552**	**conferencescheduling**	いいえ (弊社が定義)

例	ドメイン	サイズ	ディレクトリー名	コンテスト
ロックツアー	エンティティークラス 1 つ (連鎖変数 1 つ) (シャドウ変数 4 つ) シャドウエンティティークラス 1 つ (自動シャドウ変数 1 つ)	エンティティー ⇐ **47** 値 ⇐ **48** 探索空間 ⇐ **10^59**	**rocktour**	いいえ (弊社が定義)
航空機乗組員のスケジューリング	エンティティークラス 1 つ (変数 1 つ) シャドウエンティティークラス 1 つ (自動シャドウ変数 1 つ)	エンティティー ⇐ **4375** 値 ⇐ **750** 探索空間 ⇐ **10^12578**	**flightcrewscheduling**	いいえ (弊社が定義)

4.3. N クィーン

n サイズのチェスボードで、他のクィーンに取られない場所に n 個のクィーンを置きます。最も一般的な n クィーンパズルは、n = 8 の 8 個のクィーンパズルです。



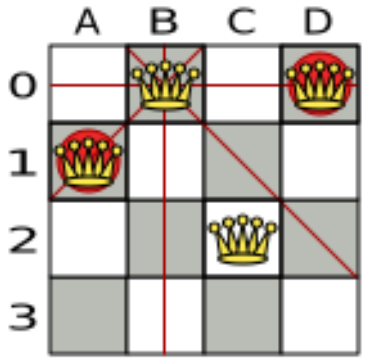
制約:

- n 列および n 行のチェスボードを使用します。
- チェスボードに n 個のクィーンを置きます。
- クィーンが他のクィーンに取られないように配置します。クィーンは、同じ水平線上、垂直線上、対角線上にある他のクィーンを取ることができます。

本書では、4 つのクイーンパズルを主な例として多用しています。

以下が提案された解です。

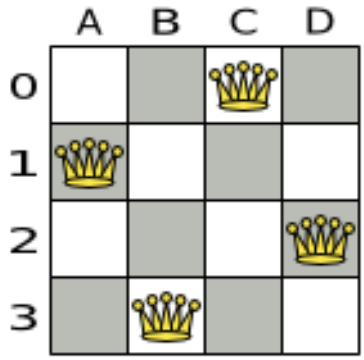
図4.1 クィーン 4 個のパズルの誤った解



上記の解は、**A1** と **B0** (および **B0** と **D0**) のクィーンがお互いに駒を取れるので間違っています。**B0** のクィーンをどかせば「他のクィーンに取られないようにする」という制約は順守できますが、「n 個のクィーンを置く」という制約に違反します。

以下は正しい解です。

図4.2 クィーン 4 個のパズルの正しい解



すべての制約が満たされているので、これが正解です。

n クィーンパズルでは、正解が複数存在する場合が多々あります。指定した n に対して考えられるすべての解を見つけるのではなく、指定の n に対する正しい解を 1 つ導き出すことに焦点をあてます。

問題の規模

```
4queens   has   4 queens with a search space of    256.
8queens   has   8 queens with a search space of   10^7.
16queens  has  16 queens with a search space of  10^19.
32queens  has  32 queens with a search space of  10^48.
64queens  has  64 queens with a search space of 10^115.
256queens has 256 queens with a search space of 10^616.
```

n クィーンは、初心者用のサンプルとして実装されているため、最適化はされていません。それにもかかわらず、クィーンが 64 個になっても簡単に処理できます。何点か変更を加えると、クィーンが 5000 個以上になっても簡単に対応できることが立証されています。

4.3.1. N クィーンのドメインモデル

この例では、4 つのクィーンの問題を解決するドメインモデルを使用します。

- ドメインモデルの作成
  適切なドメインモデルを使用すると、プランニングの問題をより簡単に理解し、解決することができます。

  以下は、n クィーンの例のドメインモデルです。

```
public class Column {

    private int index;

    // ... getters and setters
}
```

```
public class Row {

    private int index;

    // ... getters and setters
}
```

```
public class Queen {

    private Column column;
    private Row row;

    public int getAscendingDiagonalIndex() {...}
    public int getDescendingDiagonalIndex() {...}

    // ... getters and setters
}
```

- 探索空間の計算
  Queen インスタンスには **Column** (例: 0 は列 A、1 は列 B) および**Row** (例: 0 は行 0、1 は行 1) が含まれます。

  列と行をもとに、昇順の対角線、および降順の対角線を計算することができます。

  列と行のインデックスは、チェスボードの左上隅から数えています。

```
public class NQueens {

    private int n;
    private List<Column> columnList;
    private List<Row> rowList;

    private List<Queen> queenList;

    private SimpleScore score;

    // ... getters and setters
}
```

1. 解の求め方
   1 つの **NQueens** インスタンスには **Queen** インスタンスの一覧が含まれています。これが **Solution** 実装として提供され、Solver が解決して読み出します。

たとえば、4 クイーンのサンプルでは、NQueens の **getN()** メソッドが常に 4 を返します。

図4.3 クィーン 4 個の解

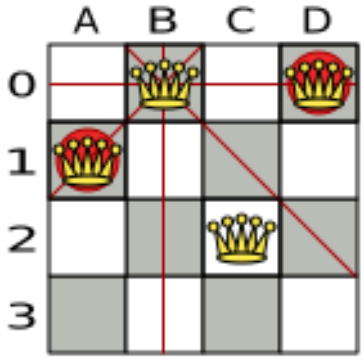


表4.2 ドメインモデルの解の詳細

	columnIndex	rowIndex	ascendingDiagonalIndex (columnIndex + rowIndex)	descendingDiagonalIndex (columnIndex - rowIndex)
A1	0	1	**1 (**)**	-1
B0	1	**0 (*)**	**1 (**)**	1
C2	2	2	4	0
D0	3	**0 (*)**	3	3

(*) や (**) のように、クィーン 2 つが同じ行、列、対角線を共有する場合は、2 つの駒が互いを取ることができます。

4.4. クラウドバランシング

この例に関する詳細は、*3章Java Solver のスタートガイド: クラウドバランシングの例*を参照してください。

4.5. 巡回セールスマン (TSP - 巡回セールスマン問題)

都市の一覧をもとに、セールスマンが最短距離で、各都市を 1 度だけ訪問するルートを探します。

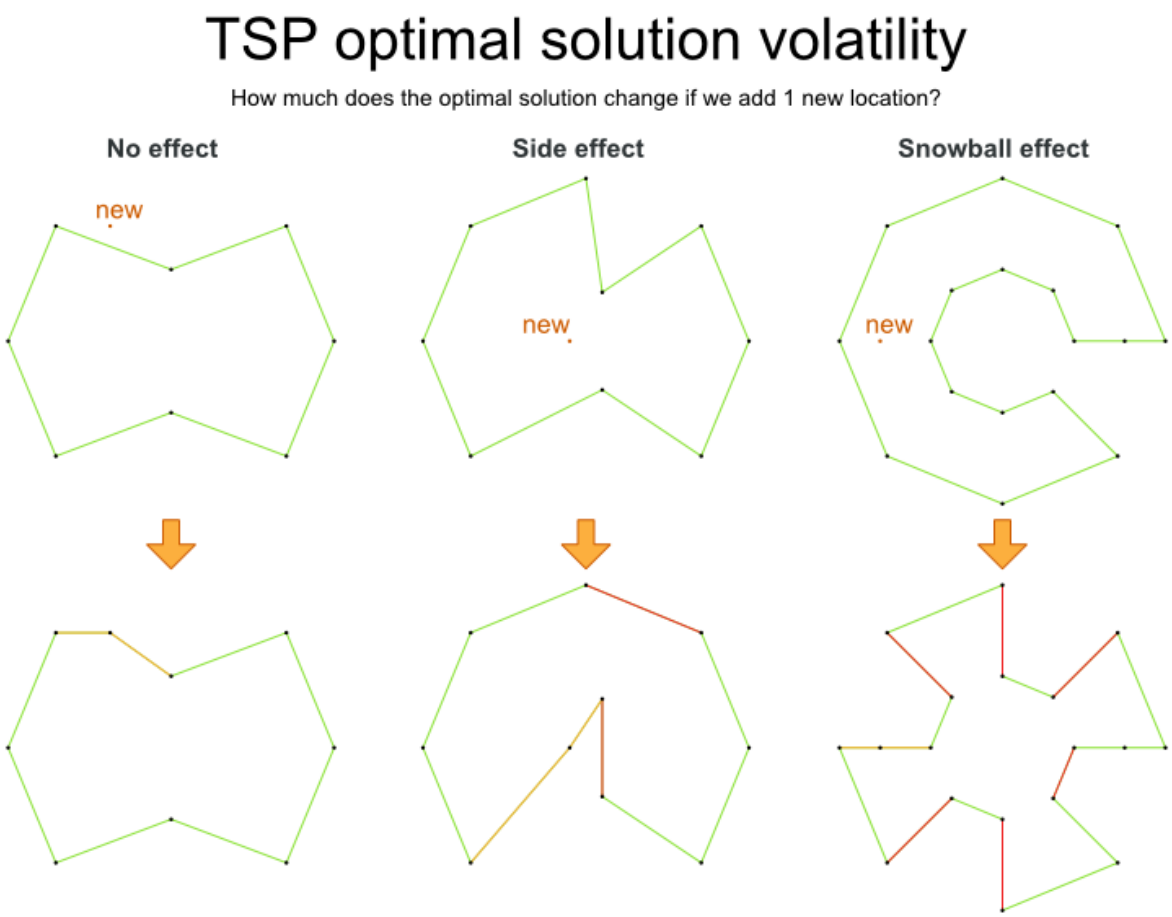
この問題は ウィキペディア に定義されています。これは、計算数学で 最も熱心に研究された問題の 1 つです。大概は、従業員のシフト勤務など、その他の制約と一緒に計画の問題の一部として使用されます。

問題の規模

```
dj38     has  38 cities with a search space of   10^43.
europe40 has  40 cities with a search space of   10^46.
st70     has  70 cities with a search space of   10^98.
pcb442   has 442 cities with a search space of  10^976.
lu980    has 980 cities with a search space of 10^2504.
```

問題の難易度

TSP の定義は単純ですが、問題の解決は驚くほど難しくなります。これは NP 困難問題と呼ばれ、多くの計画の問題と同様、特定の問題のデータセットに対する最適な解は、その問題のデータセットが少しでも変更すると、大幅に変化する可能性があります。



4.6. ディナーパーティー

マナーズさんがディナーパーティーを再度開催することにしました。

- 今回は、ゲスト 144 名を招待し、円卓を 12 個用意し、各テーブルに椅子を 12 席用意しました。
- 座席は、同じ性別の人が隣同士にならないようにします。
- また、隣の人とは必ず、同じ趣味を 1 つ持つようにします。
- 各テーブルには、政治家、医者、著名人、コーチ、教師、そしてプログラマーを 2 名ずつ配置します。
- 政治家、医者、コーチ、プログラマーは、それぞれ同じ業種の人が同じテーブルにならないようにします。

Drools Expert にも、(サイズがはるかに小さい) 一般的なミスマナーズのサンプルが含まれており、包括的なヒューリスティックを採用して解を導き出しています。Planner の実装は、ヒューリスティックを使用して最適解を見つけ、Drools Expert を使用して各解のスコアを計算するため、スケーラビリティーがはるかに高くなっています。

問題の規模

> wedding01 has 18 jobs, 144 guests, 288 hobby practicians, 12 tables and 144 seats with a search space of 10^310.

4.7. テニスクラブのスケジュール

テニスクラブでは、毎週 4 チームが総あたりで試合をします。4 つの対戦枠を公平にチームに割り当てます。

ハード制約:

- 競合: チームは 1 日に 1 回だけ試合ができる。
- 参加不可: 日程によって参加できないチームがある。

中程度の制約:

- 公平な割り当て: 各チームが試合をする回数を (ほぼ) 同じにする。

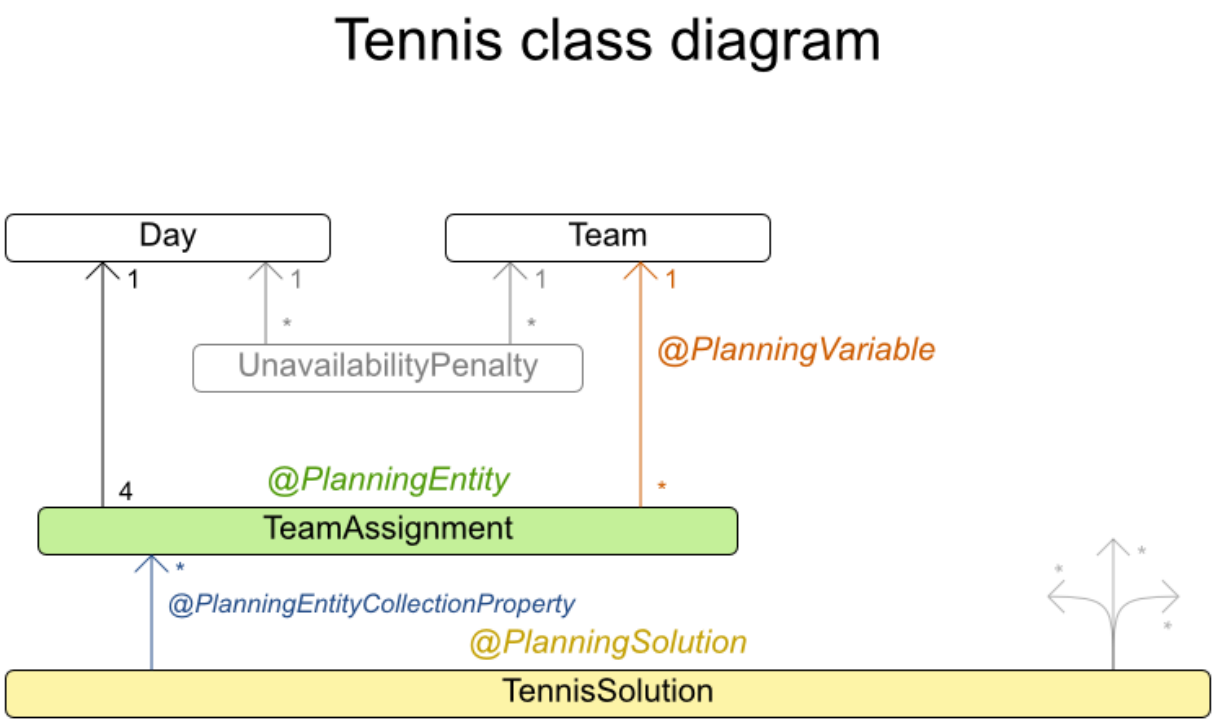
ソフト制約:

- 均等に対戦: 各チームが、各対戦相手と対戦する回数を同じにする。

問題の規模

> munich-7teams has 7 teams, 18 days, 12 unavailabilityPenalties and 72 teamAssignments with a search space of 10^60.

図4.4 ドメインモデル



4.8. 会議のスケジュール

各会議に、開始時間と会議室を割り当てます。会議の長さは異なります。

ハード制約:

- 部屋の制約: 2 つの会議が、同じ時間に同じ会議室を使用することはできない。
- 必須の出席者: 同じ時間に開催される必須の会議を 2 つ割り当てることはできない。
- 必要とされる部屋の収容人数: 会議の出席者全員を収容できない部屋では会議を行ってはいけない。
- 同日中に開始して終了: 会議は複数の日にわたってスケジュールされないようにする。

中程度の制約:

- 任意の出席者: 同じ時間に開催される任意の会議を 2 つ割り当てることはできない。また、任意の会議と必須の会議を同じ時間に割り当てることはできない。

ソフト制約:

- 早い段階でスケジュール: すべての会議をできるだけ早くスケジュールする。
- 会議と会議の間の休憩時間: 会議と会議の間には、最低でも時間枠 1 つ分、休憩を入れる必要がある。

- 会議の重複: 並行して行われる会議の数を最小限に抑えて、どちらかの会議を選択しなければならない状況をなくす。
- 先に大きい部屋から割り当てる: 参加者が登録していない場合でも、できるだけ多数の参加者を収容するために、大きい部屋が空いている場合にはその部屋から割り当てていく必要がある。
- 部屋の不変性: 会議が連続して行われ、休憩の時間枠が 2 つ分より少ない場合には、会議は同じ部屋で行う方が良い。

問題の規模

> 50meetings-160timegrains-5rooms has 50 meetings, 160 timeGrains and 5 rooms with a search space of 10^145.
> 100meetings-320timegrains-5rooms has 100 meetings, 320 timeGrains and 5 rooms with a search space of 10^320.
> 200meetings-640timegrains-5rooms has 200 meetings, 640 timeGrains and 5 rooms with a search space of 10^701.
> 400meetings-1280timegrains-5rooms has 400 meetings, 1280 timeGrains and 5 rooms with a search space of 10^1522.
> 800meetings-2560timegrains-5rooms has 800 meetings, 2560 timeGrains and 5 rooms with a search space of 10^3285.

4.9. コースの時間割 (ITC 2007 TRACK 3 - カリキュラムのスケジュール)

各授業を、時間枠および講義室に割り当ててスケジュールを組みます。

ハード制約:

- 講師の制約: 各講師は、同じ時間に授業を 2 つ受け持つことはできない。
- カリキュラムの制約: カリキュラムには、2 つの授業を同じ時間に設定することはできない。
- 部屋の占有: 同じ時間の同じ講義室に、2 つの授業を割り当てることはできない。
- 利用不可の時間 (データセットごとに指定): 授業には割り当てられない時間がある。

ソフト制約:

- 講義室の収容人数: 講義室の収容人数は、その授業を受ける学生の数よりも多くなければならない。
- 最小限の就業日数: 同じコースの授業の開講期間は、最短になるようにする。
- カリキュラムの緊密さ: 同じカリキュラムに含まれる授業は、時間帯を近く (連続した時間に) 設定する。
- 講義室の不変性: 同じコースの授業は同じ講義室を割り当てる必要がある。

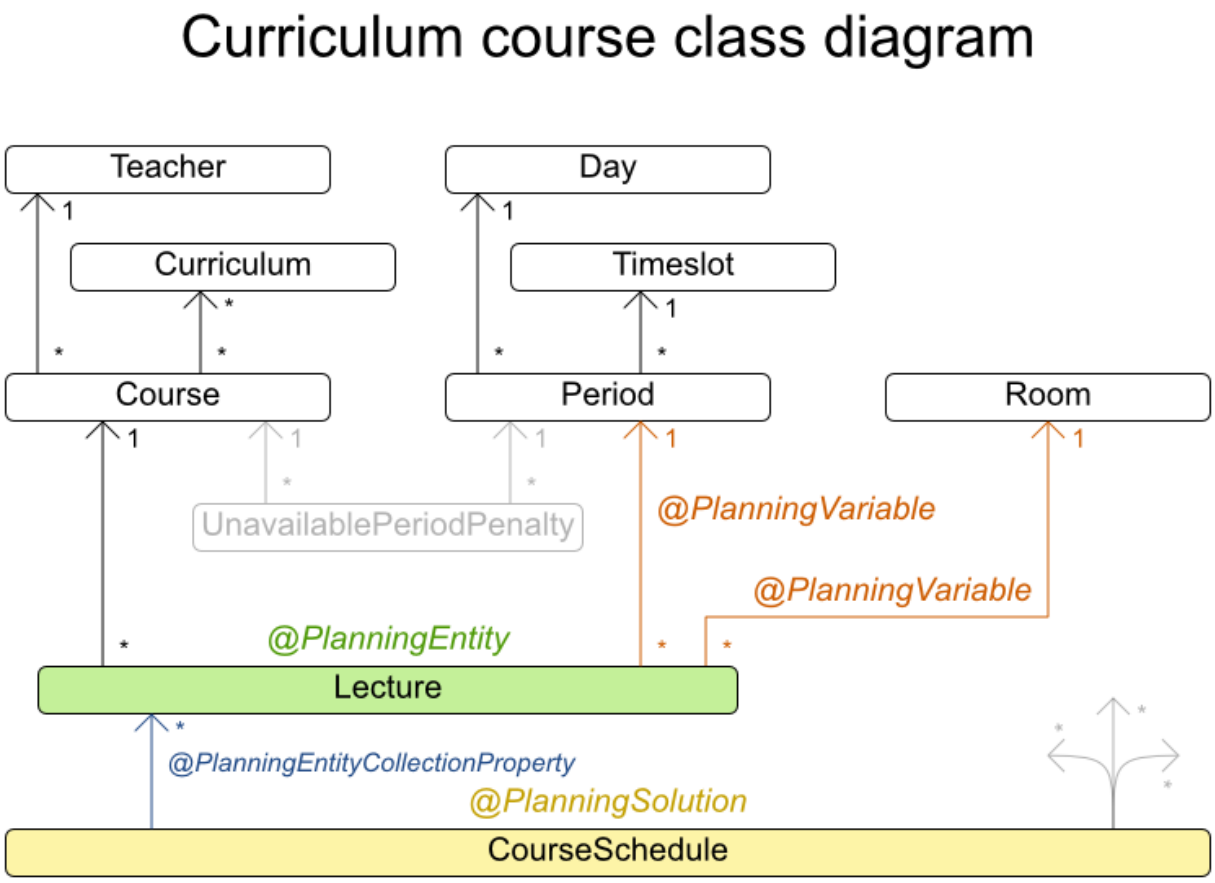
この問題は、「International Timetabling Competition 2007 track 3」で定義されています。

問題の規模

> comp01 has 24 teachers, 14 curricula, 30 courses, 160 lectures, 30 periods, 6 rooms and 53 unavailable period constraints with a search space of 10^360.
> comp02 has 71 teachers, 70 curricula, 82 courses, 283 lectures, 25 periods, 16 rooms and 513 unavailable period constraints with a search space of 10^736.

comp03 has 61 teachers, 68 curricula, 72 courses, 251 lectures, 25 periods, 16 rooms and 382 unavailable period constraints with a search space of 10^653.
comp04 has 70 teachers, 57 curricula, 79 courses, 286 lectures, 25 periods, 18 rooms and 396 unavailable period constraints with a search space of 10^758.
comp05 has 47 teachers, 139 curricula, 54 courses, 152 lectures, 36 periods, 9 rooms and 771 unavailable period constraints with a search space of 10^381.
comp06 has 87 teachers, 70 curricula, 108 courses, 361 lectures, 25 periods, 18 rooms and 632 unavailable period constraints with a search space of 10^957.
comp07 has 99 teachers, 77 curricula, 131 courses, 434 lectures, 25 periods, 20 rooms and 667 unavailable period constraints with a search space of 10^1171.
comp08 has 76 teachers, 61 curricula, 86 courses, 324 lectures, 25 periods, 18 rooms and 478 unavailable period constraints with a search space of 10^859.
comp09 has 68 teachers, 75 curricula, 76 courses, 279 lectures, 25 periods, 18 rooms and 405 unavailable period constraints with a search space of 10^740.
comp10 has 88 teachers, 67 curricula, 115 courses, 370 lectures, 25 periods, 18 rooms and 694 unavailable period constraints with a search space of 10^981.
comp11 has 24 teachers, 13 curricula, 30 courses, 162 lectures, 45 periods, 5 rooms and 94 unavailable period constraints with a search space of 10^381.
comp12 has 74 teachers, 150 curricula, 88 courses, 218 lectures, 36 periods, 11 rooms and 1368 unavailable period constraints with a search space of 10^566.
comp13 has 77 teachers, 66 curricula, 82 courses, 308 lectures, 25 periods, 19 rooms and 468 unavailable period constraints with a search space of 10^824.
comp14 has 68 teachers, 60 curricula, 85 courses, 275 lectures, 25 periods, 17 rooms and 486 unavailable period constraints with a search space of 10^722.

図4.5 ドメインモデル



4.10. マシンの再割当て (GOOGLE ROADEF 2012)

各プロセスをマシンに割り当てます。全プロセスには、すでに元の (最適化されていない) 割り当てがあります。プロセスにはそれぞれ、各リソース (CPU、メモリーなど) が一定量必要です。これは、クラウドのバランスの例の応用です。

ハード制約:

- 最大容量: マシンに割り当てる各リソースはこの量を超えてはいけない。
- 競合: 同じサービスのプロセスは別のマシンで実行する必要がある。
- 分散: 同じサービスのプロセスは複数の場所に分散させる必要がある。
- 依存関係: 他のサービスに依存するサービスのプロセスは、そのサービスの近くで実行する必要がある。
- 一時的な使用: リソースによっては一時的なものがあり、元のマシンと、新たに割り当てられたマシンの両方の最大容量にカウントされる。

ソフト制約:

- 負荷: 各マシンの各リソースの安全容量を超えてはいけない。
- 負荷分散: 各マシンで利用可能なリソースを分散させて、今後の割り当てに対応できるように容量を空ける。
- プロセスの移動コスト: プロセスには移動コストが発生する。
- サービスの移動コスト: サービスには移動コストが発生する。
- 機械の移動コスト: マシン A からマシン B にプロセスを移動すると、A から B に固有の移動コストが別途発生する。

この問題は the Google ROADEF/EURO Challenge 2012 で定義されています。

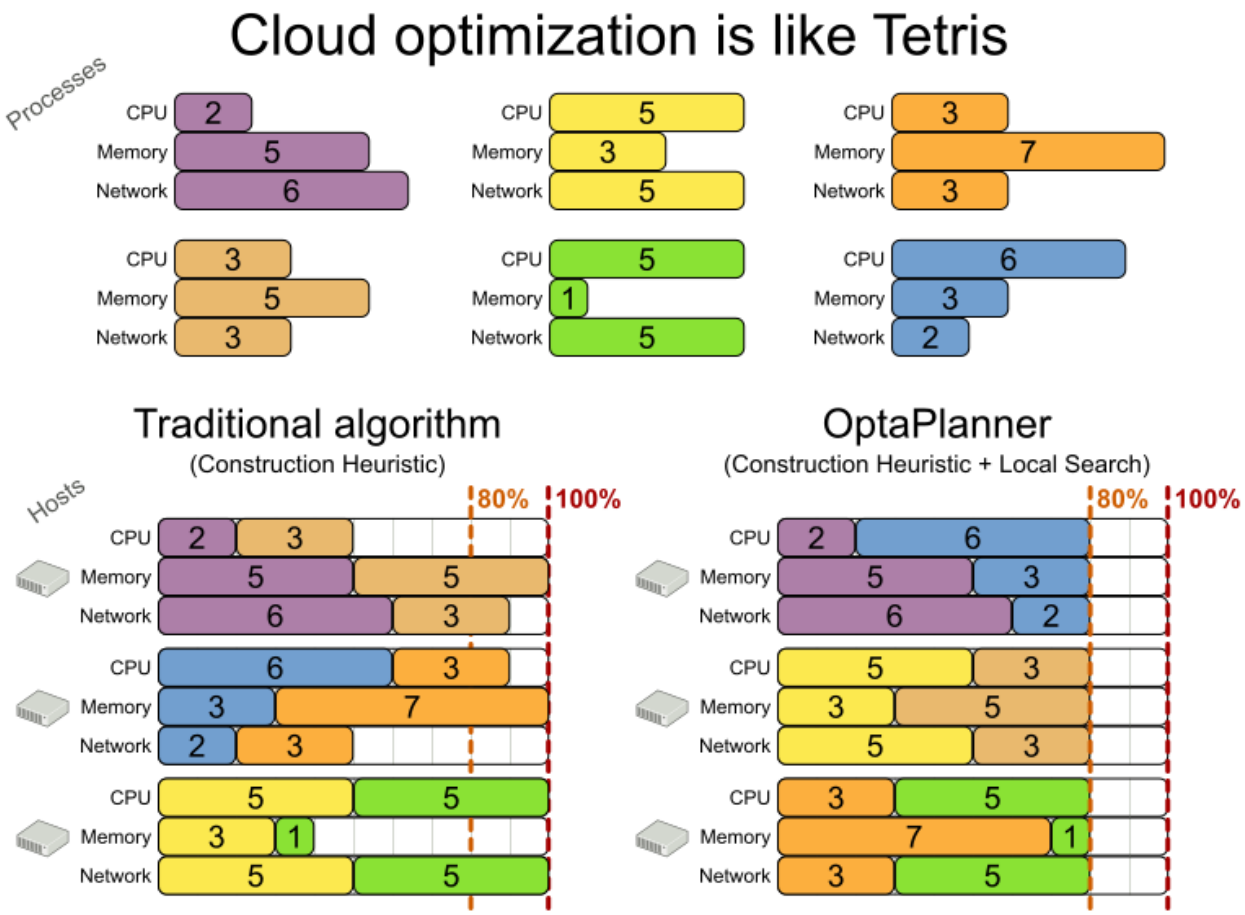
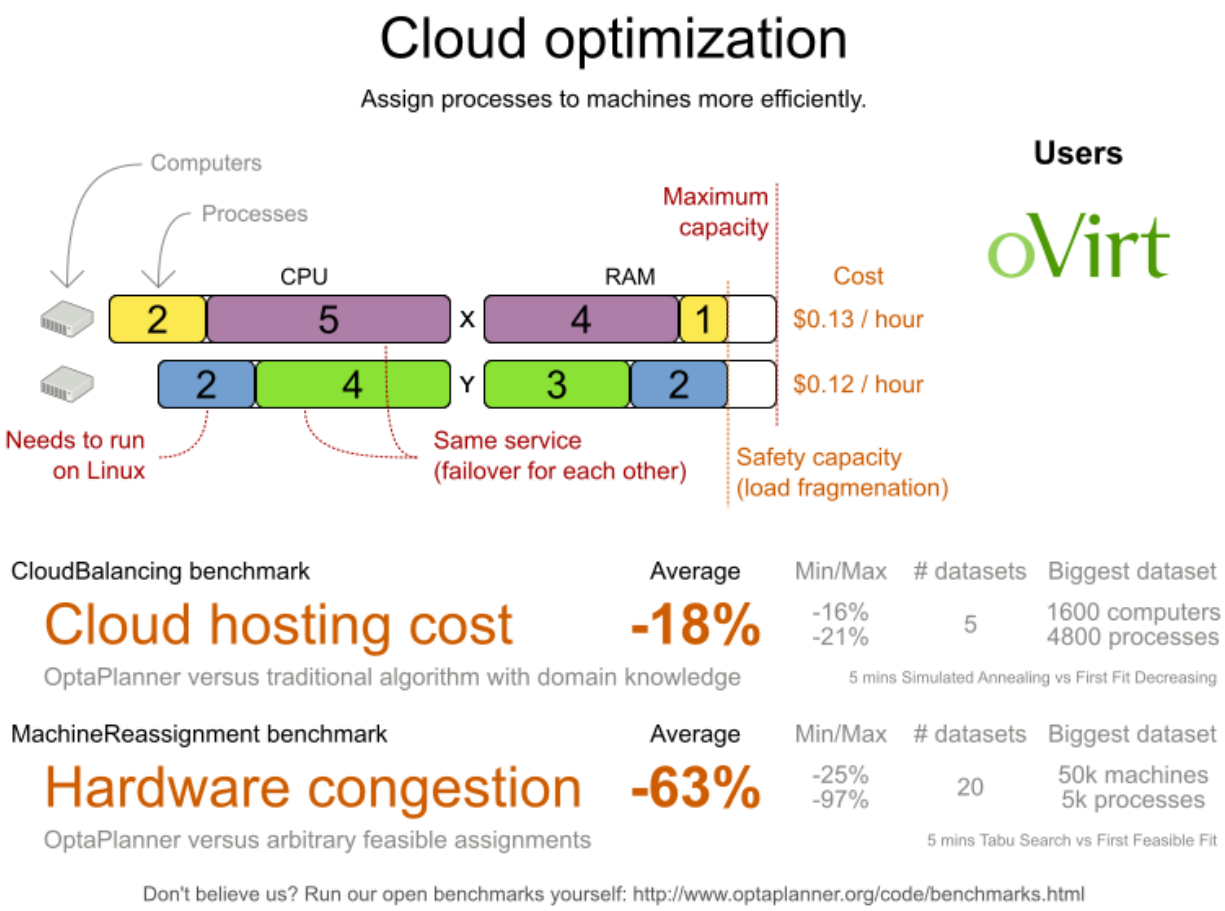

Cloud optimization is like Tetris
Processes
CPU 2
Memory 5
Network 6
CPU 5
Memory 3
Network 5
CPU 3
Memory 7
Network 3
CPU 3
Memory 5
Network 3
CPU 5
Memory 1
Network 5
CPU 6
Memory 3
Network 2
Traditional algorithm
(Construction Heuristic)
Hosts
80%
100%
CPU 2 3
Memory 5 5
Network 6 3
CPU 6 3
Memory 3 7
Network 2 3
CPU 5 5
Memory 3 1
Network 5 5
OptaPlanner
(Construction Heuristic + Local Search)
80%
100%
CPU 2 6
Memory 5 3
Network 6 2
CPU 5 3
Memory 3 5
Network 5 3
CPU 3 5
Memory 7 1
Network 3 5

図4.6 価値提案

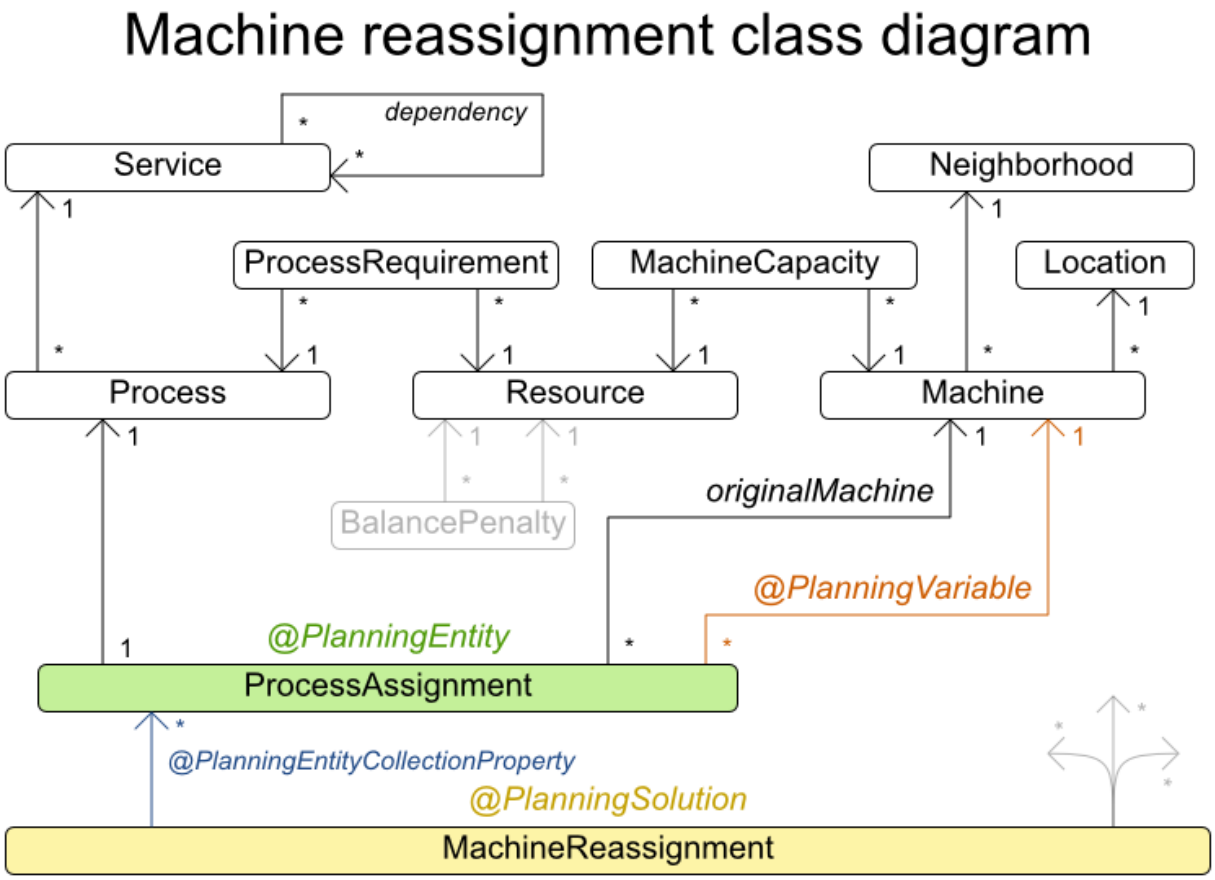


問題の規模

```
model_a1_1 has  2 resources,  1 neighborhoods,   4 locations,    4 machines,    79 services,   100
processes and 1 balancePenalties with a search space of     10^60.
model_a1_2 has  4 resources,  2 neighborhoods,   4 locations,  100 machines,   980 services,  1000
processes and 0 balancePenalties with a search space of   10^2000.
model_a1_3 has  3 resources,  5 neighborhoods,  25 locations,  100 machines,   216 services,  1000
processes and 0 balancePenalties with a search space of   10^2000.
model_a1_4 has  3 resources, 50 neighborhoods,  50 locations,   50 machines,   142 services,  1000
processes and 1 balancePenalties with a search space of   10^1698.
model_a1_5 has  4 resources,  2 neighborhoods,   4 locations,   12 machines,   981 services,  1000
processes and 1 balancePenalties with a search space of   10^1079.
model_a2_1 has  3 resources,  1 neighborhoods,   1 locations,  100 machines,  1000 services,  1000
processes and 0 balancePenalties with a search space of   10^2000.
model_a2_2 has 12 resources,  5 neighborhoods,  25 locations,  100 machines,   170 services,  1000
processes and 0 balancePenalties with a search space of   10^2000.
model_a2_3 has 12 resources,  5 neighborhoods,  25 locations,  100 machines,   129 services,  1000
processes and 0 balancePenalties with a search space of   10^2000.
model_a2_4 has 12 resources,  5 neighborhoods,  25 locations,   50 machines,   180 services,  1000
processes and 1 balancePenalties with a search space of   10^1698.
model_a2_5 has 12 resources,  5 neighborhoods,  25 locations,   50 machines,   153 services,  1000
processes and 0 balancePenalties with a search space of   10^1698.
model_b_1  has 12 resources,  5 neighborhoods,  10 locations,  100 machines,  2512 services,  5000
processes and 0 balancePenalties with a search space of  10^10000.
model_b_2  has 12 resources,  5 neighborhoods,  10 locations,  100 machines,  2462 services,  5000
processes and 1 balancePenalties with a search space of  10^10000.
```

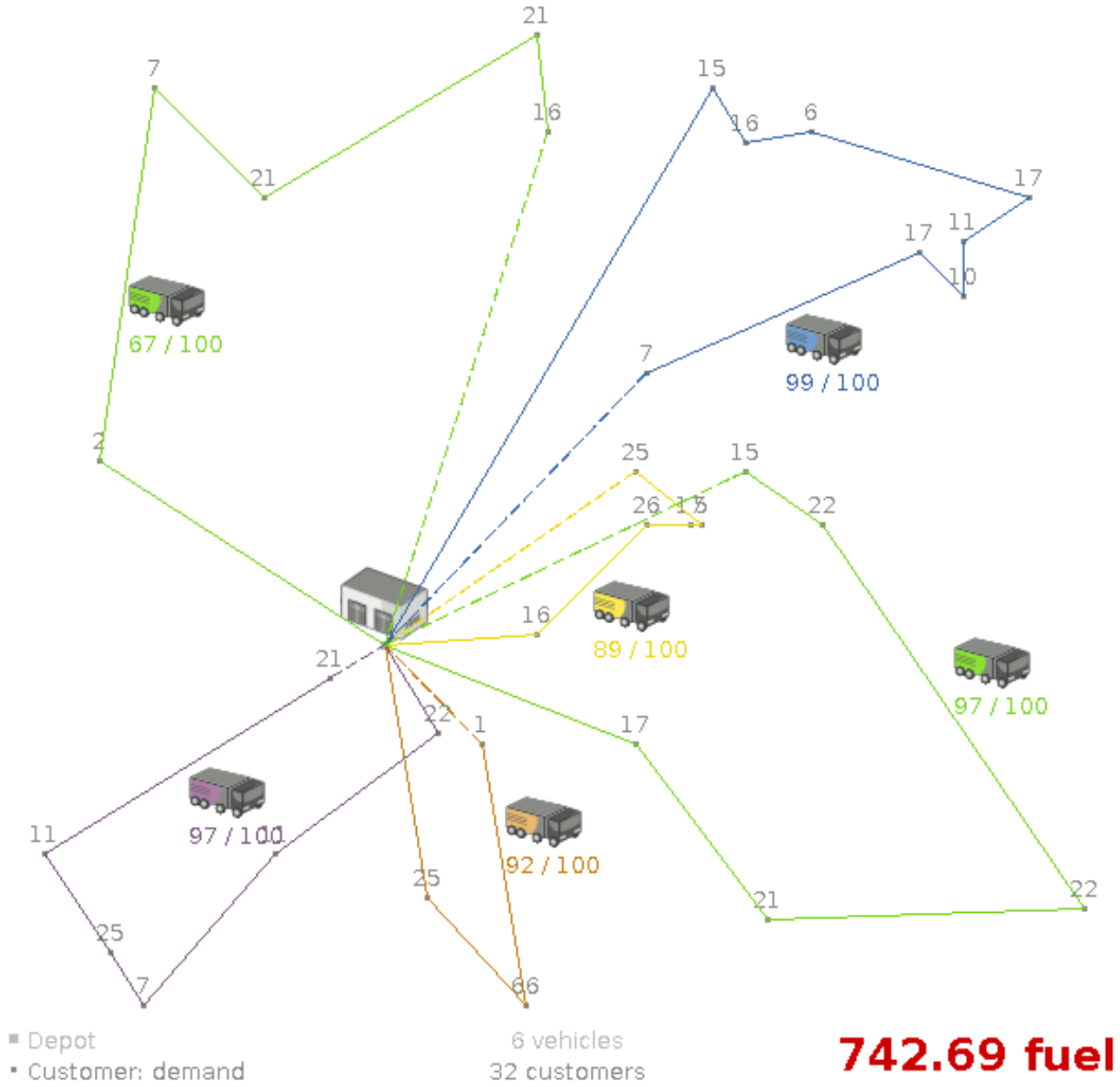
model_b_3 has 6 resources, 5 neighborhoods, 10 locations, 100 machines, 15025 services, 20000 processes and 0 balancePenalties with a search space of 10^40000.
model_b_4 has 6 resources, 5 neighborhoods, 50 locations, 500 machines, 1732 services, 20000 processes and 1 balancePenalties with a search space of 10^53979.
model_b_5 has 6 resources, 5 neighborhoods, 10 locations, 100 machines, 35082 services, 40000 processes and 0 balancePenalties with a search space of 10^80000.
model_b_6 has 6 resources, 5 neighborhoods, 50 locations, 200 machines, 14680 services, 40000 processes and 1 balancePenalties with a search space of 10^92041.
model_b_7 has 6 resources, 5 neighborhoods, 50 locations, 4000 machines, 15050 services, 40000 processes and 1 balancePenalties with a search space of 10^144082.
model_b_8 has 3 resources, 5 neighborhoods, 10 locations, 100 machines, 45030 services, 50000 processes and 0 balancePenalties with a search space of 10^100000.
model_b_9 has 3 resources, 5 neighborhoods, 100 locations, 1000 machines, 4609 services, 50000 processes and 1 balancePenalties with a search space of 10^150000.
model_b_10 has 3 resources, 5 neighborhoods, 100 locations, 5000 machines, 4896 services, 50000 processes and 1 balancePenalties with a search space of 10^184948.

図4.7 ドメインモデル



4.11. 配送経路

複数の車両を使用して、各顧客の品物を回収し、倉庫まで運びます。1つの車両で複数の顧客から品物を回収することはできますが、収容できる容量には限りがあります。



基本例 (CVRP) のほかに、時間枠の設定が加わった例 (CVRPTW) もあります。

ハード制約:

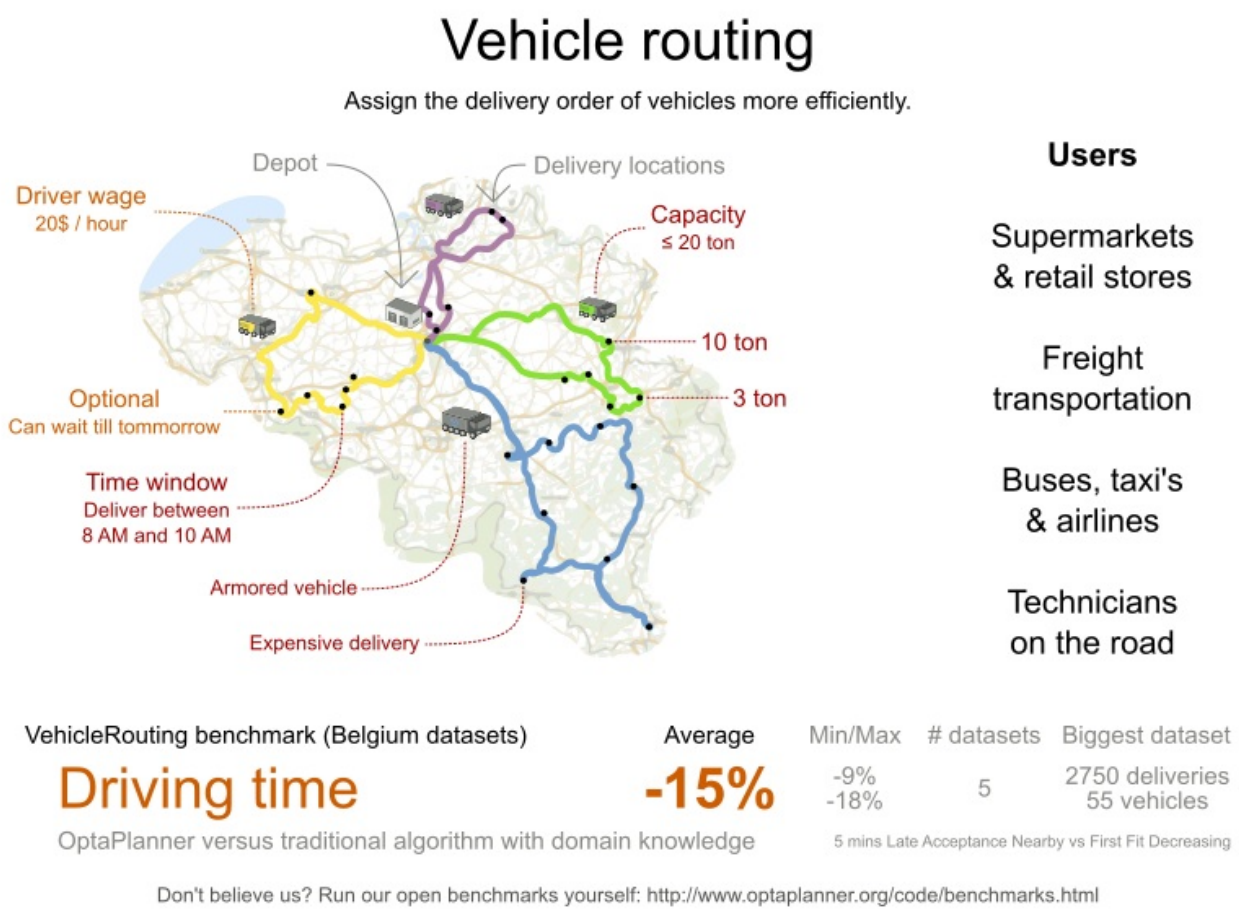
- 車両の容量: 車両は、車載容量を超えて品物を運ぶことができない。
- 時間枠 (CVRPTW のみ):
  - 移動時間: 別の場所に移動する場合には時間がかかる。
  - 顧客対応の時間: 車両は顧客に対応している時間、顧客先にとどまる必要がある。
  - 顧客の準備が整う時間: 顧客の準備が整う前に車両が到着する可能性があるが、準備ができるまで待機してから顧客に対応する必要がある。
  - 顧客が設定した締め切り時間: 車両は、顧客が設定した締め切り時間までに到着する必要がある。

ソフト制約:

- 合計距離: 車両が移動する合計距離 (ガソリンの消費量) を最小限に抑える。

CVRP (Capacitated Vehicle Routing Problem) と CVRPTW (Capacitated Vehicle Routing Problem with Time Window) は、「VRP Web」で定義されています。

図4.8 価値提案



問題の規模

CVRP インスタンス (時間枠なし):

```
belgium-n50-k10           has  1 depots, 10 vehicles and   49 customers with a search space of 10^74.
belgium-n100-k10          has  1 depots, 10 vehicles and   99 customers with a search space of 10^170.
belgium-n500-k20          has  1 depots, 20 vehicles and  499 customers with a search space of 10^1168.
belgium-n1000-k20         has  1 depots, 20 vehicles and  999 customers with a search space of 10^2607.
belgium-n2750-k55         has  1 depots, 55 vehicles and 2749 customers with a search space of 10^8380.
belgium-road-km-n50-k10   has  1 depots, 10 vehicles and   49 customers with a search space of 10^74.
belgium-road-km-n100-k10  has  1 depots, 10 vehicles and   99 customers with a search space of 10^170.
belgium-road-km-n500-k20  has  1 depots, 20 vehicles and  499 customers with a search space of 10^1168.
belgium-road-km-n1000-k20 has  1 depots, 20 vehicles and  999 customers with a search space of 10^2607.
belgium-road-km-n2750-k55 has  1 depots, 55 vehicles and 2749 customers with a search space of
```

```
10^8380.
belgium-road-time-n50-k10   has  1 depots, 10 vehicles and   49 customers with a search space of
10^74.
belgium-road-time-n100-k10  has  1 depots, 10 vehicles and   99 customers with a search space of
10^170.
belgium-road-time-n500-k20  has  1 depots, 20 vehicles and  499 customers with a search space of
10^1168.
belgium-road-time-n1000-k20 has  1 depots, 20 vehicles and  999 customers with a search space of
10^2607.
belgium-road-time-n2750-k55 has  1 depots, 55 vehicles and 2749 customers with a search space of
10^8380.
belgium-d2-n50-k10          has  2 depots, 10 vehicles and   48 customers with a search space of
10^74.
belgium-d3-n100-k10         has  3 depots, 10 vehicles and   97 customers with a search space of
10^170.
belgium-d5-n500-k20         has  5 depots, 20 vehicles and  495 customers with a search space of
10^1168.
belgium-d8-n1000-k20        has  8 depots, 20 vehicles and  992 customers with a search space of
10^2607.
belgium-d10-n2750-k55       has 10 depots, 55 vehicles and 2740 customers with a search space of
10^8380.

A-n32-k5  has 1 depots,  5 vehicles and  31 customers with a search space of  10^40.
A-n33-k5  has 1 depots,  5 vehicles and  32 customers with a search space of  10^41.
A-n33-k6  has 1 depots,  6 vehicles and  32 customers with a search space of  10^42.
A-n34-k5  has 1 depots,  5 vehicles and  33 customers with a search space of  10^43.
A-n36-k5  has 1 depots,  5 vehicles and  35 customers with a search space of  10^46.
A-n37-k5  has 1 depots,  5 vehicles and  36 customers with a search space of  10^48.
A-n37-k6  has 1 depots,  6 vehicles and  36 customers with a search space of  10^49.
A-n38-k5  has 1 depots,  5 vehicles and  37 customers with a search space of  10^49.
A-n39-k5  has 1 depots,  5 vehicles and  38 customers with a search space of  10^51.
A-n39-k6  has 1 depots,  6 vehicles and  38 customers with a search space of  10^52.
A-n44-k7  has 1 depots,  7 vehicles and  43 customers with a search space of  10^61.
A-n45-k6  has 1 depots,  6 vehicles and  44 customers with a search space of  10^62.
A-n45-k7  has 1 depots,  7 vehicles and  44 customers with a search space of  10^63.
A-n46-k7  has 1 depots,  7 vehicles and  45 customers with a search space of  10^65.
A-n48-k7  has 1 depots,  7 vehicles and  47 customers with a search space of  10^68.
A-n53-k7  has 1 depots,  7 vehicles and  52 customers with a search space of  10^77.
A-n54-k7  has 1 depots,  7 vehicles and  53 customers with a search space of  10^79.
A-n55-k9  has 1 depots,  9 vehicles and  54 customers with a search space of  10^82.
A-n60-k9  has 1 depots,  9 vehicles and  59 customers with a search space of  10^91.
A-n61-k9  has 1 depots,  9 vehicles and  60 customers with a search space of  10^93.
A-n62-k8  has 1 depots,  8 vehicles and  61 customers with a search space of  10^94.
A-n63-k9  has 1 depots,  9 vehicles and  62 customers with a search space of  10^97.
A-n63-k10 has 1 depots, 10 vehicles and  62 customers with a search space of  10^98.
A-n64-k9  has 1 depots,  9 vehicles and  63 customers with a search space of  10^99.
A-n65-k9  has 1 depots,  9 vehicles and  64 customers with a search space of 10^101.
A-n69-k9  has 1 depots,  9 vehicles and  68 customers with a search space of 10^108.
A-n80-k10 has 1 depots, 10 vehicles and  79 customers with a search space of 10^130.
F-n45-k4  has 1 depots,  4 vehicles and  44 customers with a search space of  10^60.
F-n72-k4  has 1 depots,  4 vehicles and  71 customers with a search space of 10^108.
F-n135-k7 has 1 depots,  7 vehicles and 134 customers with a search space of 10^240.
```

CVRPTW インスタンス (時間枠あり):

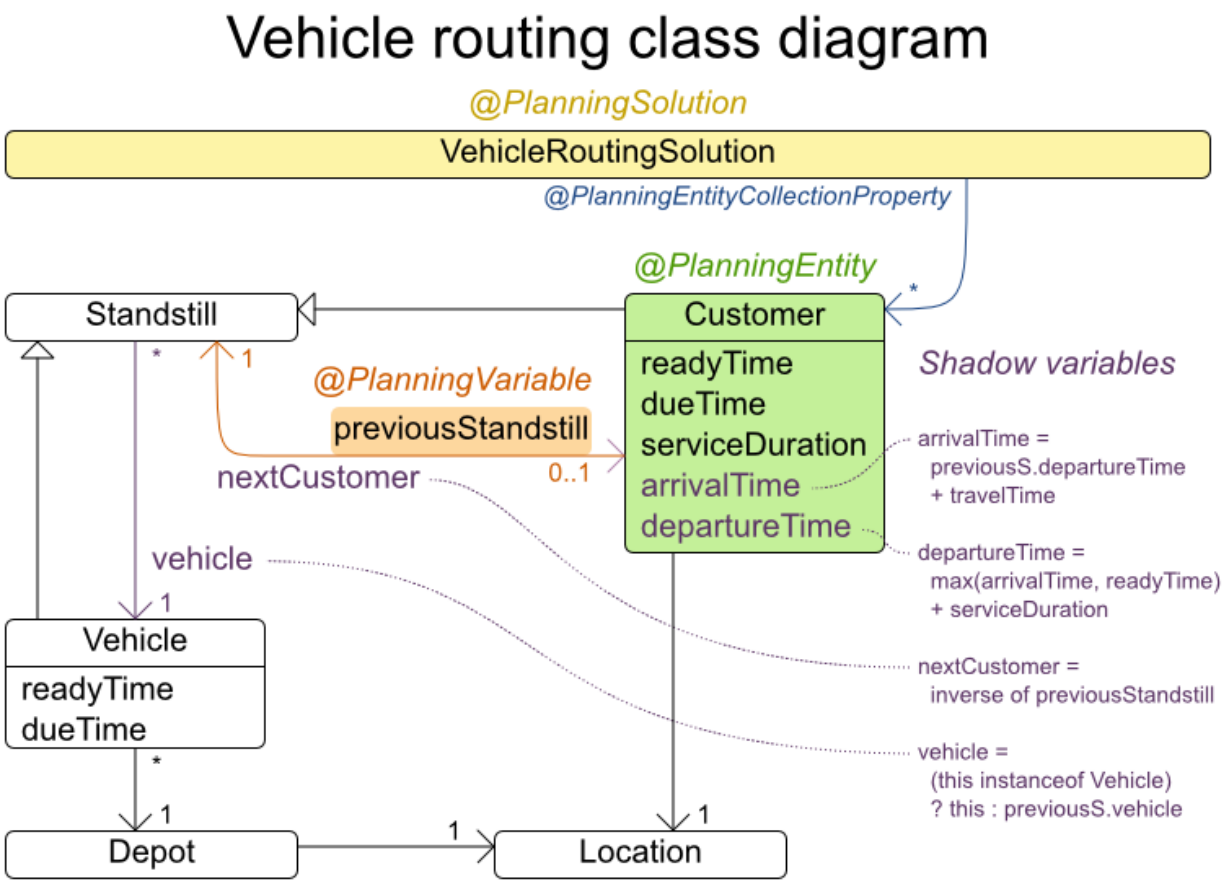
```
belgium-tw-d2-n50-k10    has  2 depots, 10 vehicles and   48 customers with a search space of
```

10^74.
belgium-tw-d3-n100-k10 has 3 depots, 10 vehicles and 97 customers with a search space of 10^170.
belgium-tw-d5-n500-k20 has 5 depots, 20 vehicles and 495 customers with a search space of 10^1168.
belgium-tw-d8-n1000-k20 has 8 depots, 20 vehicles and 992 customers with a search space of 10^2607.
belgium-tw-d10-n2750-k55 has 10 depots, 55 vehicles and 2740 customers with a search space of 10^8380.
belgium-tw-n50-k10 has 1 depots, 10 vehicles and 49 customers with a search space of 10^74.
belgium-tw-n100-k10 has 1 depots, 10 vehicles and 99 customers with a search space of 10^170.
belgium-tw-n500-k20 has 1 depots, 20 vehicles and 499 customers with a search space of 10^1168.
belgium-tw-n1000-k20 has 1 depots, 20 vehicles and 999 customers with a search space of 10^2607.
belgium-tw-n2750-k55 has 1 depots, 55 vehicles and 2749 customers with a search space of 10^8380.

Solomon_025_C101 has 1 depots, 25 vehicles and 25 customers with a search space of 10^40.
Solomon_025_C201 has 1 depots, 25 vehicles and 25 customers with a search space of 10^40.
Solomon_025_R101 has 1 depots, 25 vehicles and 25 customers with a search space of 10^40.
Solomon_025_R201 has 1 depots, 25 vehicles and 25 customers with a search space of 10^40.
Solomon_025_RC101 has 1 depots, 25 vehicles and 25 customers with a search space of 10^40.
Solomon_025_RC201 has 1 depots, 25 vehicles and 25 customers with a search space of 10^40.
Solomon_100_C101 has 1 depots, 25 vehicles and 100 customers with a search space of 10^185.
Solomon_100_C201 has 1 depots, 25 vehicles and 100 customers with a search space of 10^185.
Solomon_100_R101 has 1 depots, 25 vehicles and 100 customers with a search space of 10^185.
Solomon_100_R201 has 1 depots, 25 vehicles and 100 customers with a search space of 10^185.
Solomon_100_RC101 has 1 depots, 25 vehicles and 100 customers with a search space of 10^185.
Solomon_100_RC201 has 1 depots, 25 vehicles and 100 customers with a search space of 10^185.
Homberger_0200_C1_2_1 has 1 depots, 50 vehicles and 200 customers with a search space of 10^429.
Homberger_0200_C2_2_1 has 1 depots, 50 vehicles and 200 customers with a search space of 10^429.
Homberger_0200_R1_2_1 has 1 depots, 50 vehicles and 200 customers with a search space of 10^429.
Homberger_0200_R2_2_1 has 1 depots, 50 vehicles and 200 customers with a search space of 10^429.
Homberger_0200_RC1_2_1 has 1 depots, 50 vehicles and 200 customers with a search space of 10^429.
Homberger_0200_RC2_2_1 has 1 depots, 50 vehicles and 200 customers with a search space of 10^429.

```
Homberger_0400_C1_4_1  has 1 depots, 100 vehicles and  400 customers with a search space of 10^978.
Homberger_0400_C2_4_1  has 1 depots, 100 vehicles and  400 customers with a search space of 10^978.
Homberger_0400_R1_4_1  has 1 depots, 100 vehicles and  400 customers with a search space of 10^978.
Homberger_0400_R2_4_1  has 1 depots, 100 vehicles and  400 customers with a search space of 10^978.
Homberger_0400_RC1_4_1 has 1 depots, 100 vehicles and  400 customers with a search space of 10^978.
Homberger_0400_RC2_4_1 has 1 depots, 100 vehicles and  400 customers with a search space of 10^978.
Homberger_0600_C1_6_1  has 1 depots, 150 vehicles and  600 customers with a search space of 10^1571.
Homberger_0600_C2_6_1  has 1 depots, 150 vehicles and  600 customers with a search space of 10^1571.
Homberger_0600_R1_6_1  has 1 depots, 150 vehicles and  600 customers with a search space of 10^1571.
Homberger_0600_R2_6_1  has 1 depots, 150 vehicles and  600 customers with a search space of 10^1571.
Homberger_0600_RC1_6_1 has 1 depots, 150 vehicles and  600 customers with a search space of 10^1571.
Homberger_0600_RC2_6_1 has 1 depots, 150 vehicles and  600 customers with a search space of 10^1571.
Homberger_0800_C1_8_1  has 1 depots, 200 vehicles and  800 customers with a search space of 10^2195.
Homberger_0800_C2_8_1  has 1 depots, 200 vehicles and  800 customers with a search space of 10^2195.
Homberger_0800_R1_8_1  has 1 depots, 200 vehicles and  800 customers with a search space of 10^2195.
Homberger_0800_R2_8_1  has 1 depots, 200 vehicles and  800 customers with a search space of 10^2195.
Homberger_0800_RC1_8_1 has 1 depots, 200 vehicles and  800 customers with a search space of 10^2195.
Homberger_0800_RC2_8_1 has 1 depots, 200 vehicles and  800 customers with a search space of 10^2195.
Homberger_1000_C110_1  has 1 depots, 250 vehicles and 1000 customers with a search space of 10^2840.
Homberger_1000_C210_1  has 1 depots, 250 vehicles and 1000 customers with a search space of 10^2840.
Homberger_1000_R110_1  has 1 depots, 250 vehicles and 1000 customers with a search space of 10^2840.
Homberger_1000_R210_1  has 1 depots, 250 vehicles and 1000 customers with a search space of 10^2840.
Homberger_1000_RC110_1 has 1 depots, 250 vehicles and 1000 customers with a search space of 10^2840.
Homberger_1000_RC210_1 has 1 depots, 250 vehicles and 1000 customers with a search space of 10^2840.
```

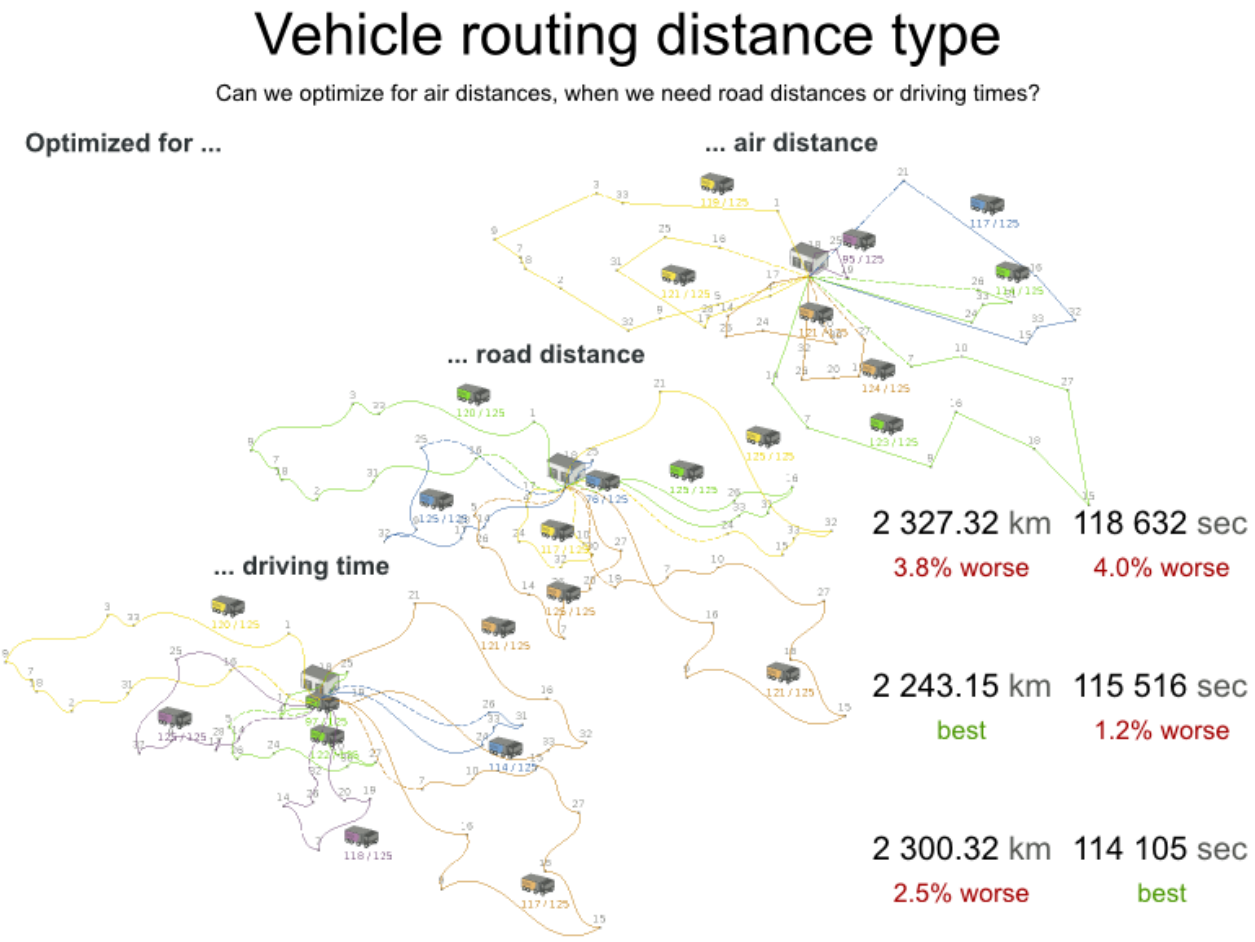
4.11.1. 配送経路のドメインモデル



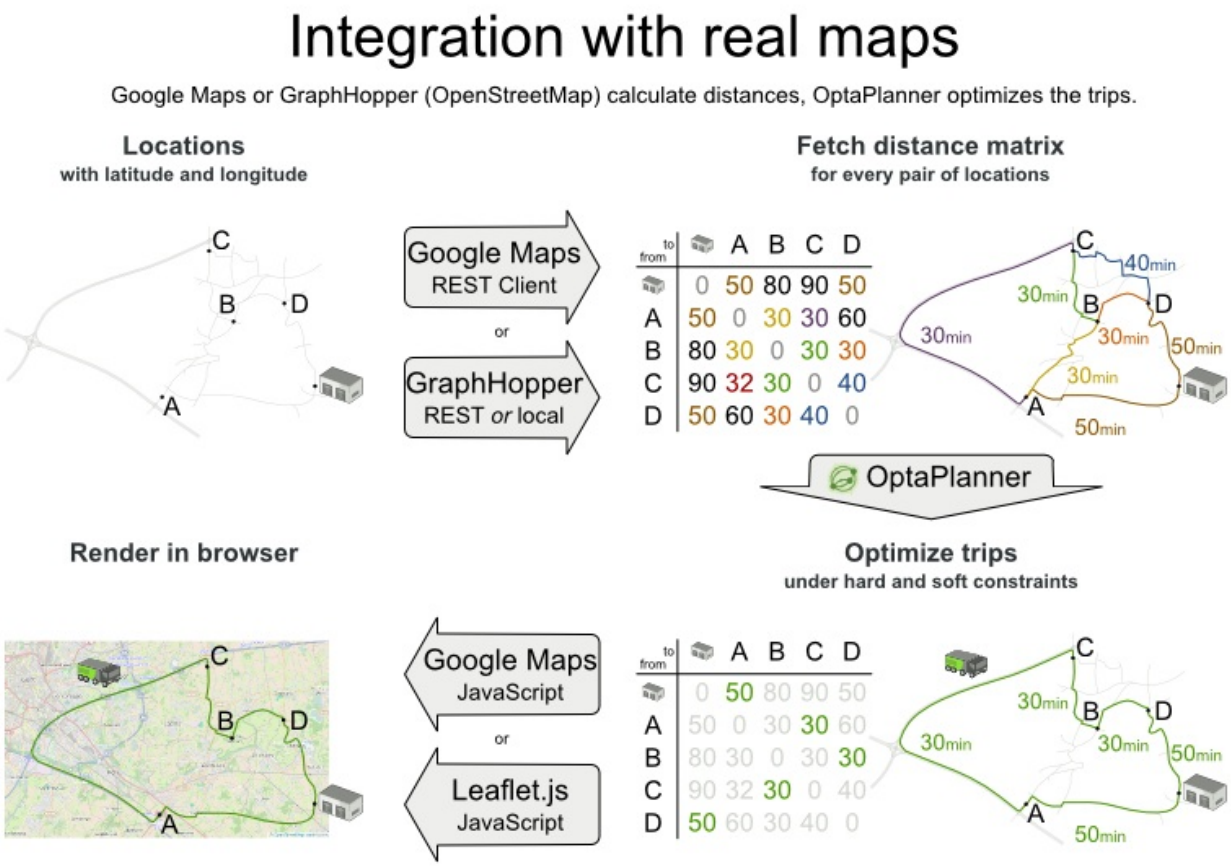
時間枠ありの配送経路のドメインモデルでは、シャドウ変数の機能を多用します。こうすることで、**arrivalTime** や **departureTime** などのプロパティーがドメインモデルで直接利用できるため、制約をより自然に表現できます。

直線距離ではなく道路の距離

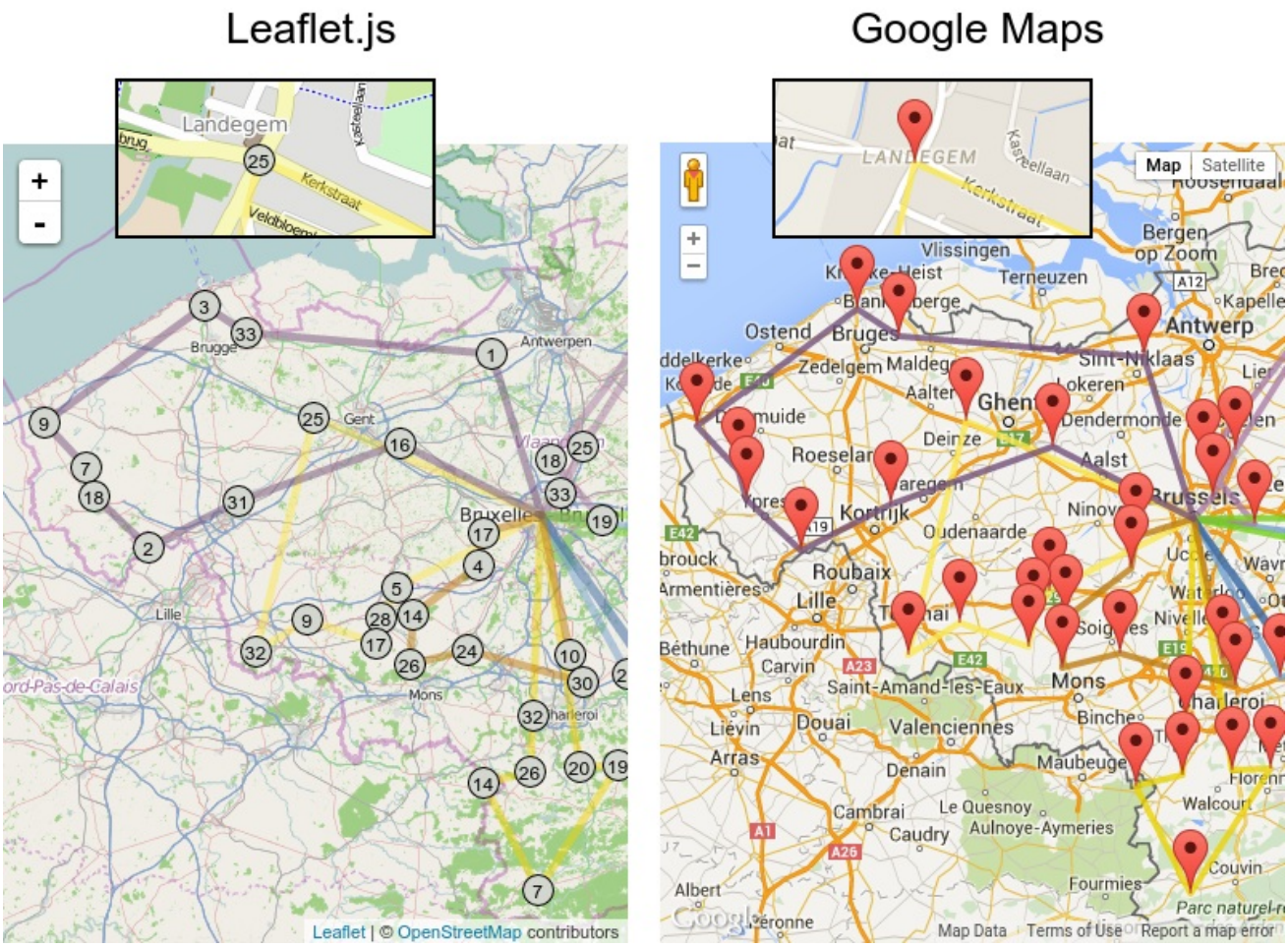
車は、直線距離を移動するのではなく、道路や高速道路を使用する必要があります。ビジネスの観点からすると、これは非常に重要です。



最適化アルゴリズムでは、2 点の距離を検索できている (できれば、事前に計算されている) 場合には、これは特に重要ではありません。移動のコストについては、距離の代わりに移動時間、ガソリン代、またはこれらの重み関数をベースにすることも可能です。GraphHopper (埋め込み可能なオフライン Java エンジン)、Open MapQuest (web サービス)、Google Maps Client API (web サービス) など、移動コストを事前に計算する技術があります。

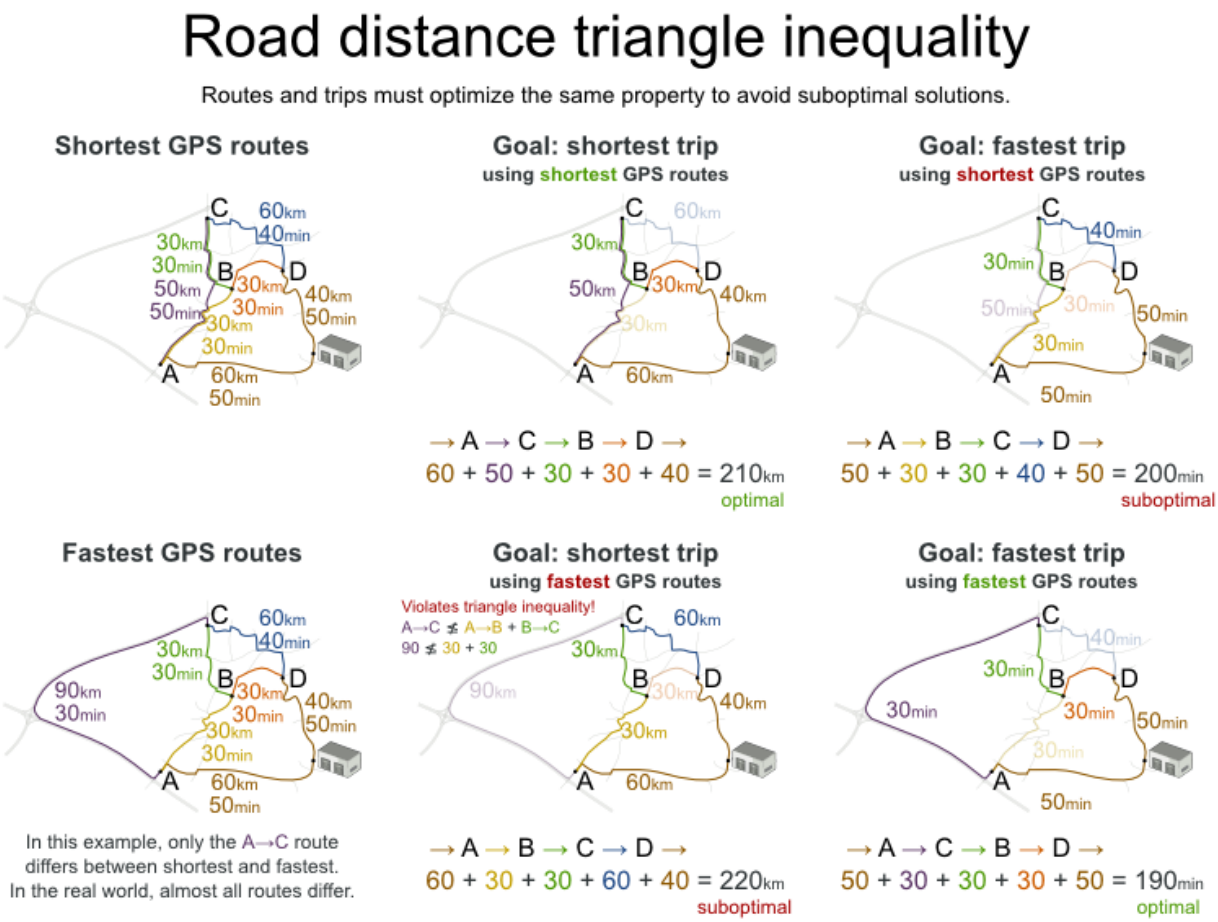


また、Leaflet や Google Maps for developers など、移動コストをレンダリングする技術もあります。**optaplanner-webexamples-*.war** には、このようなレンダリングのデモ例が含まれています。



GraphHopper または Google Map Directions を使用して実際の経路をレンダリングすることも可能ですが、高速道路で経路が重なるため、停止する順番を確認するのが困難になります。

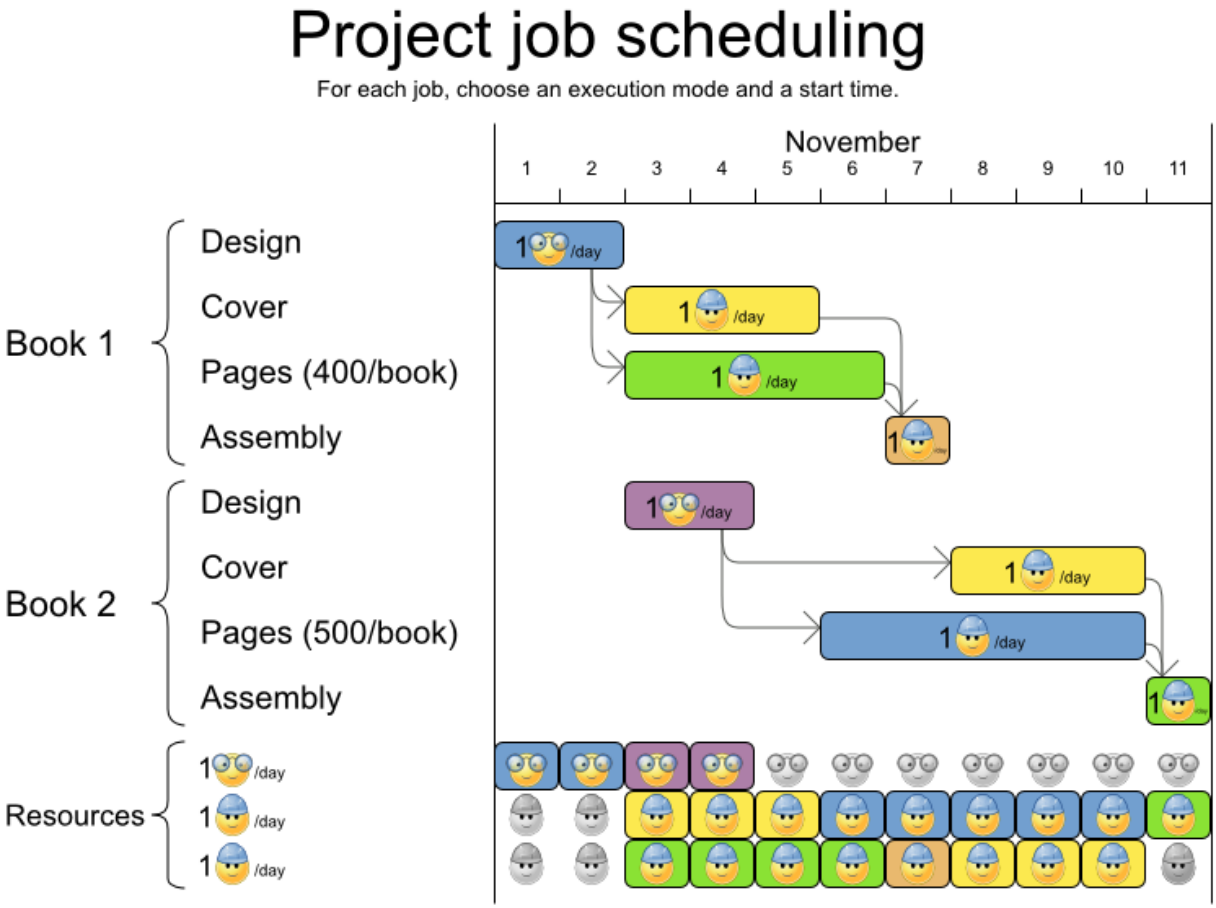
2 点間の移動コストは、**Planner** で使用したのと同じ最適化条件を使用する点に注意してください。たとえば、**GraphHopper** はデフォルトで、最短ではなく、最速の経路を返します。「最速」の **GPS** 経路の **km** (またはマイル) の距離を使用して、**Planner** で「最短」の移動を最適化しないようにしてください。以下のように、準最適な解が導き出される可能性があります。



一般的な考え方とは異なり、多くのユーザーは最短の経路ではなく、最速の経路を使用したいと考えます。通常の道路よりも高速道路の使用を好みます。舗装されていない道よりも舗装されている道路を好みます。実際には、最速の経路と、最短の経路が同じであることはほとんどありません。

4.12. プロジェクトジョブのスケジュール

プロジェクトの遅延を最小限に抑えるために、すべてのジョブを時間内に実行できるようにスケジュールを設定します。各ジョブは、プロジェクトに含まれます。ジョブは、異なる方法で実行できます。方法ごとに期間や使用するリソースが異なります。これは、柔軟な ジョブショップスケジューリング (JSP) の応用です。



ハード制約:

- ジョブの優先順位: ジョブは、先行のジョブがすべて完了するまで開始しない。
- リソースの容量: 利用可能な量を超えるリソースを使用しない。
  - リソースはローカル (同じプロジェクトのジョブ間で共有)、またはグローバル (全ジョブ間で共有) とする。
  - リソースは更新可能 (1 日に利用可能な容量) または更新不可 (全日で利用可能な容量) とする。

中程度の制約:

- プロジェクトの合計遅延時間: 各プロジェクトの所要時間 (メイクスパン) を最短にする。

ソフト制約:

- メイクスパン合計: 複数のプロジェクトスケジュールの合計所要時間を最短にする。

この問題は、「the MISTA 2013 challenge」で定義されています。

問題の規模

> Schedule A-1 has 2 projects, 24 jobs, 64 execution modes, 7 resources and 150 resource requirements.
> Schedule A-2 has 2 projects, 44 jobs, 124 execution modes, 7 resources and 420 resource requirements.

```
Schedule A-3  has  2 projects,  64 jobs,  184 execution modes,  7 resources and  630 resource requirements.
Schedule A-4  has  5 projects,  60 jobs,  160 execution modes, 16 resources and  390 resource requirements.
Schedule A-5  has  5 projects, 110 jobs,  310 execution modes, 16 resources and  900 resource requirements.
Schedule A-6  has  5 projects, 160 jobs,  460 execution modes, 16 resources and 1440 resource requirements.
Schedule A-7  has 10 projects, 120 jobs,  320 execution modes, 22 resources and  900 resource requirements.
Schedule A-8  has 10 projects, 220 jobs,  620 execution modes, 22 resources and 1860 resource requirements.
Schedule A-9  has 10 projects, 320 jobs,  920 execution modes, 31 resources and 2880 resource requirements.
Schedule A-10 has 10 projects, 320 jobs,  920 execution modes, 31 resources and 2970 resource requirements.
Schedule B-1  has 10 projects, 120 jobs,  320 execution modes, 31 resources and  900 resource requirements.
Schedule B-2  has 10 projects, 220 jobs,  620 execution modes, 22 resources and 1740 resource requirements.
Schedule B-3  has 10 projects, 320 jobs,  920 execution modes, 31 resources and 3060 resource requirements.
Schedule B-4  has 15 projects, 180 jobs,  480 execution modes, 46 resources and 1530 resource requirements.
Schedule B-5  has 15 projects, 330 jobs,  930 execution modes, 46 resources and 2760 resource requirements.
Schedule B-6  has 15 projects, 480 jobs, 1380 execution modes, 46 resources and 4500 resource requirements.
Schedule B-7  has 20 projects, 240 jobs,  640 execution modes, 61 resources and 1710 resource requirements.
Schedule B-8  has 20 projects, 440 jobs, 1240 execution modes, 42 resources and 3180 resource requirements.
Schedule B-9  has 20 projects, 640 jobs, 1840 execution modes, 61 resources and 5940 resource requirements.
Schedule B-10 has 20 projects, 460 jobs, 1300 execution modes, 42 resources and 4260 resource requirements.
```

4.13. タスクの割り当て

従業員のキューのスポットに各タスクを割り当てます。タスクごとに、従業員のアフィニティーレベルから影響を受ける期間と、タスクの顧客が含まれます。

ハード制約:

- スキル: タスクごとに 1 つ以上のスキルが必要である。従業員には、このようなスキルがすべて必要です。

ソフトレベル 0 の制約:

- 極めて重要なタスク: 主要なタスクやマイナーなタスクの前に、極めて重要なタスクを完了する。

ソフトレベル 1 の制約:

- メークスパンの最小化: 全タスクを完了するまでの時間を短縮する。
    - 勤務歴の長い従業員から順番に進めていき、公平性やロードバランシングを作成する

- 勤務歴の長い従業員から順番に進めていき、公平性やロードバランシングを作成する。

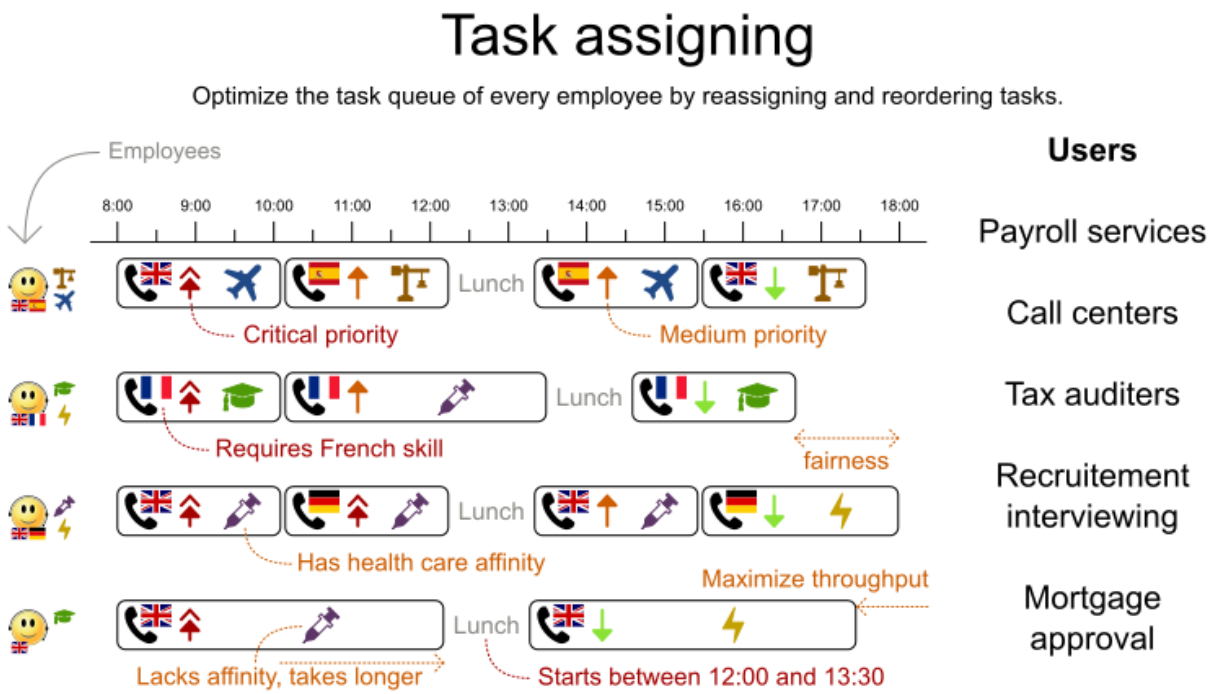
ソフトレベル 2 の制約:

- 主要なタスク: マイナーなタスクの前に、主要なタスクをできるだけ早く完了する。

ソフトレベル 3 の制約:

- マイナーなタスク: できるだけ早くマイナーなタスクを完了する。

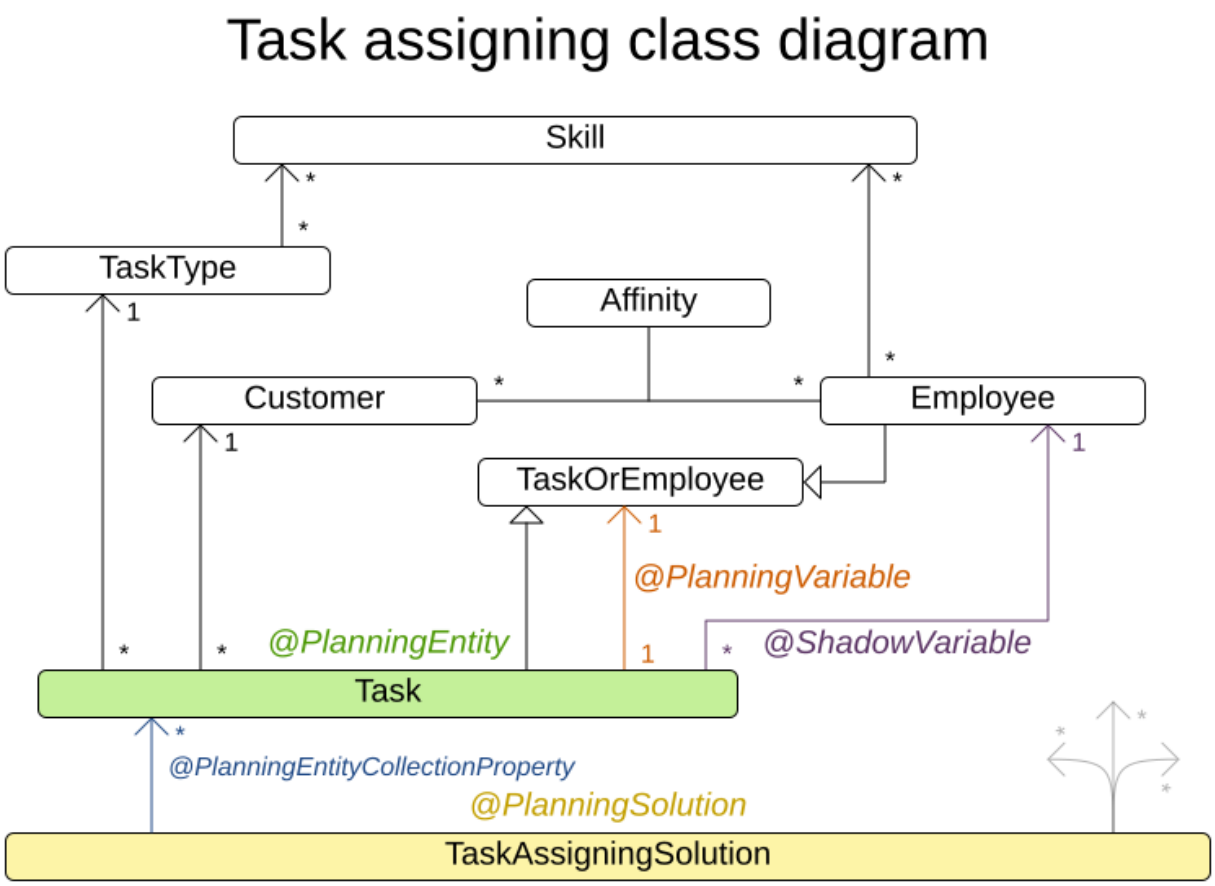
図4.9 価値提案



問題の規模

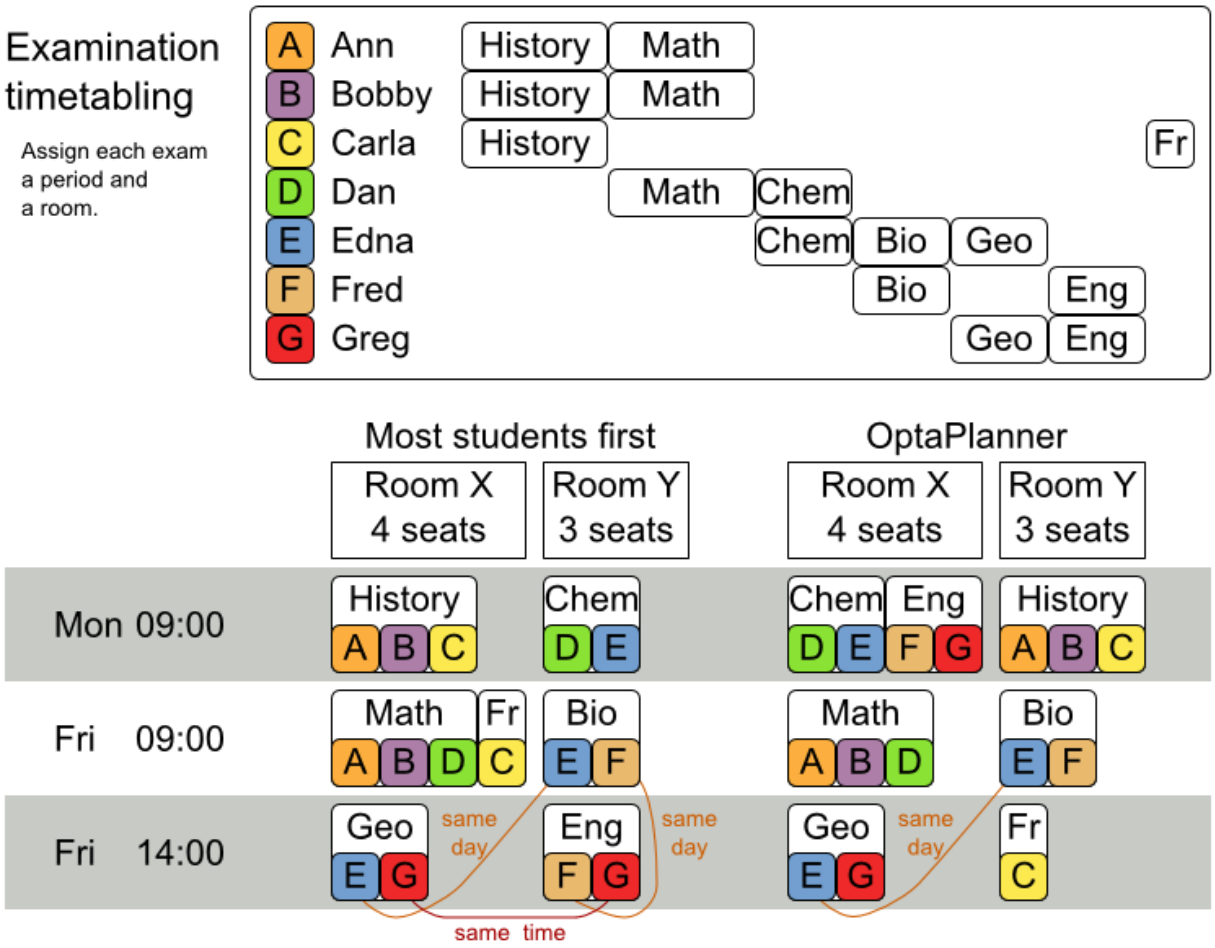
> 24tasks-8employees has 24 tasks, 6 skills, 8 employees, 4 task types and 4 customers with a search space of 10^30.
> 50tasks-5employees has 50 tasks, 5 skills, 5 employees, 10 task types and 10 customers with a search space of 10^69.
> 100tasks-5employees has 100 tasks, 5 skills, 5 employees, 20 task types and 15 customers with a search space of 10^164.
> 500tasks-20employees has 500 tasks, 6 skills, 20 employees, 100 task types and 60 customers with a search space of 10^1168.

図4.10 ドメインモデル



4.14. 試験の時間割 (ITC 2007 TRACK 1 – 試験)

すべての試験に、時間と部屋を割り当てます。同じ時間帯に同じ部屋で、複数の試験を行うことができるものとします。



ハード制約:

- 試験の制約: 同じ学生が受ける 2 つの試験は、同じ時間帯に実施できないものとする。
- 教室の収容人数: 教室の座席数は、常に受験者数よりも多くなければならない。
- 期間: 期間は、すべての試験に対応できる長さでなければならない。
- 期間関連のハード制約 (データセットごとに指定):
  - 一致: 特定の 2 つの試験を同じ時間帯に設定する必要がある (別の教室を使用することも可能)。
  - 除外: 特定の 2 つの試験を同じ時間帯に設定できない。
  - 以降: 特定の試験を、別の特定の試験の後に行う必要がある。
- 教室関連の制約 (データセットごとに指定):
  - 排他的: 特定の試験を、他の試験と同じ教室で行うことはできない。

ソフト制約 (パラメーター化されたペナルティーがそれぞれ設定されている):

- 同じ学生が、続けて試験を 2 つ受けてはいけない。
- 同じ学生が、同じ日に試験を 2 つ受けてはいけない。
- 時間帯の分散: 同じ学生が受ける 2 つの試験は、時間をある程度あける。

- 異なる試験の長さ: 教室を共有する 2 つの試験の長さは、同じにする。
- 前倒し: 規模の大きい試験は、スケジュールを早めに決定する。
- 期間のペナルティー (データセットごとに指定): 期間によっては、使用されるとペナルティーが発生する。
- 部屋のペナルティー (データセットごとに指定): 部屋によっては、使用されるとペナルティーが発生する。

実際に大学から取得した大規模な試験データセットを使用します。

この問題は、「International Timetabling Competition 2007 track 1」で定義されています。**Geoffrey De Smet** 氏は、ごく初期段階の **Planner** を使用して、このコンペティションで 4 位を獲得しました。このコンペティション以降、多くの改良点が加えられています。

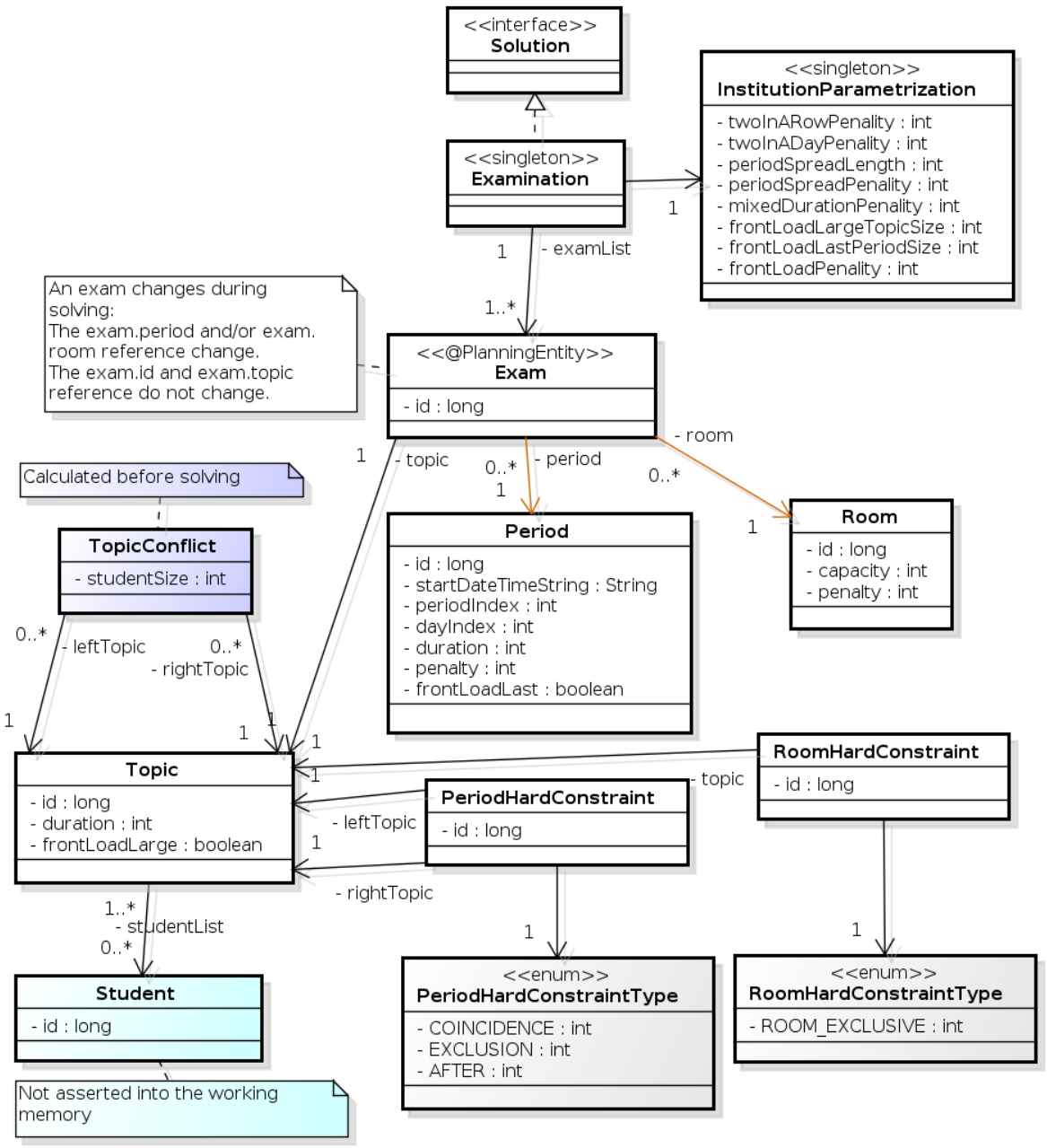
問題の規模

```
exam_comp_set1 has  7883 students,  607 exams, 54 periods,  7 rooms,  12 period constraints and
0 room constraints with a search space of 10^1564.
exam_comp_set2 has 12484 students,  870 exams, 40 periods, 49 rooms,  12 period constraints and
2 room constraints with a search space of 10^2864.
exam_comp_set3 has 16365 students,  934 exams, 36 periods, 48 rooms, 168 period constraints and
15 room constraints with a search space of 10^3023.
exam_comp_set4 has  4421 students,  273 exams, 21 periods,  1 rooms,  40 period constraints and
0 room constraints with a search space of  10^360.
exam_comp_set5 has  8719 students, 1018 exams, 42 periods,  3 rooms,  27 period constraints and
0 room constraints with a search space of 10^2138.
exam_comp_set6 has  7909 students,  242 exams, 16 periods,  8 rooms,  22 period constraints and
0 room constraints with a search space of  10^509.
exam_comp_set7 has 13795 students, 1096 exams, 80 periods, 15 rooms,  28 period constraints and
0 room constraints with a search space of 10^3374.
exam_comp_set8 has  7718 students,  598 exams, 80 periods,  8 rooms,  20 period constraints and
1 room constraints with a search space of 10^1678.
```

4.14.1. 試験の時間割のドメインモデル

以下の図では、主な試験のドメインクラスを紹介しています。

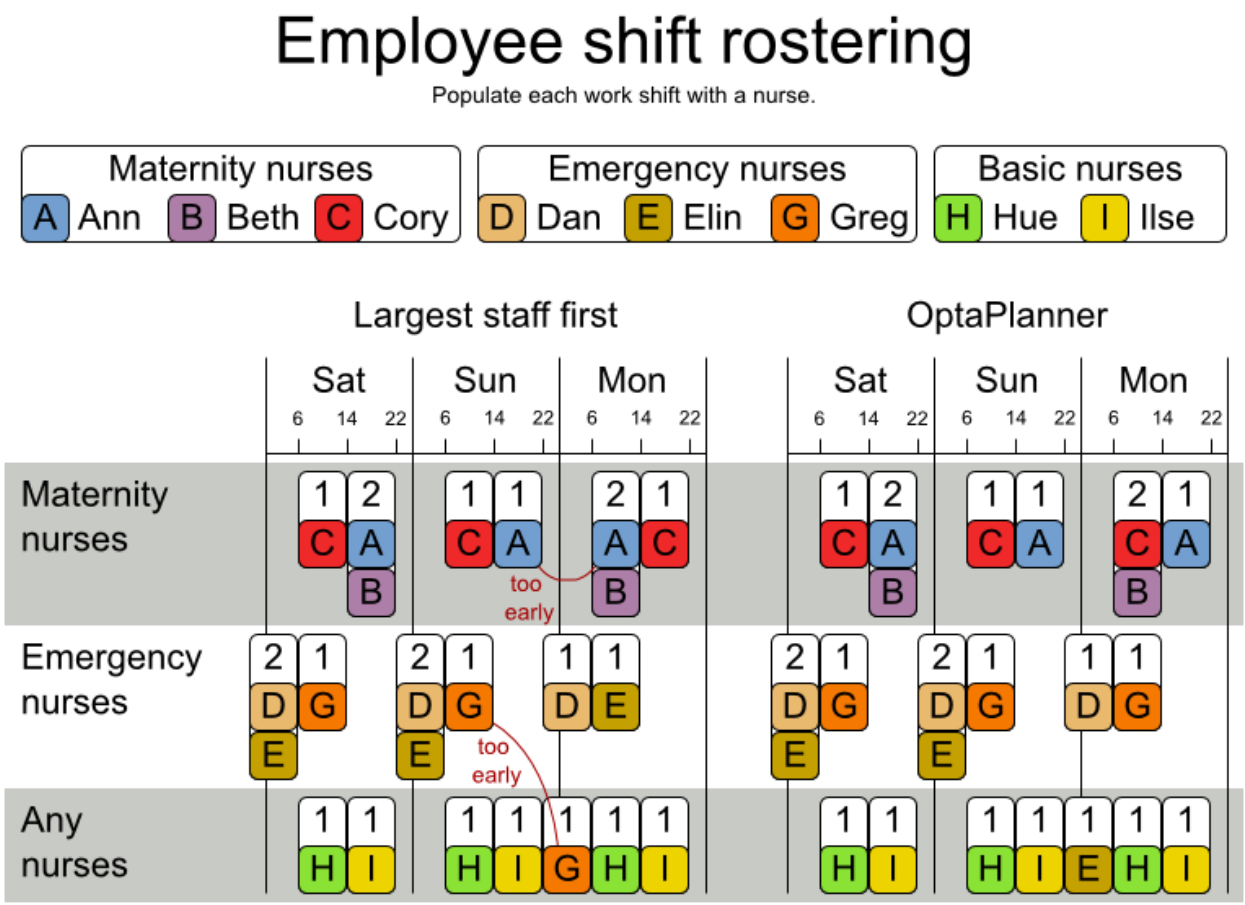
図4.11 試験のドメインクラスの図



試験のコンセプトを、**Exam** クラスと **Topic** クラスに分けた点に注意してください。期間または教室のプロパティーを変更し、解 (プランニングエンティティークラス) を求めると、**Exam** インスタンスが変化します。このとき、**Topic** インスタンス、**Period** インスタンス、および **Room** インスタンスは変化しません (他のクラスと同様、これらも問題ファクトです)。

4.15. 看護師の勤務表 (INRC 2010)

各シフトに看護師を割り当てます。



ハード制約:

- 未割り当てのシフトなし (組み込み): すべてのシフトを従業員に割り当てる必要がある。
- シフトの制約: 従業員には 1 日に 1 シフトだけ割り当てることができる。

ソフト制約:

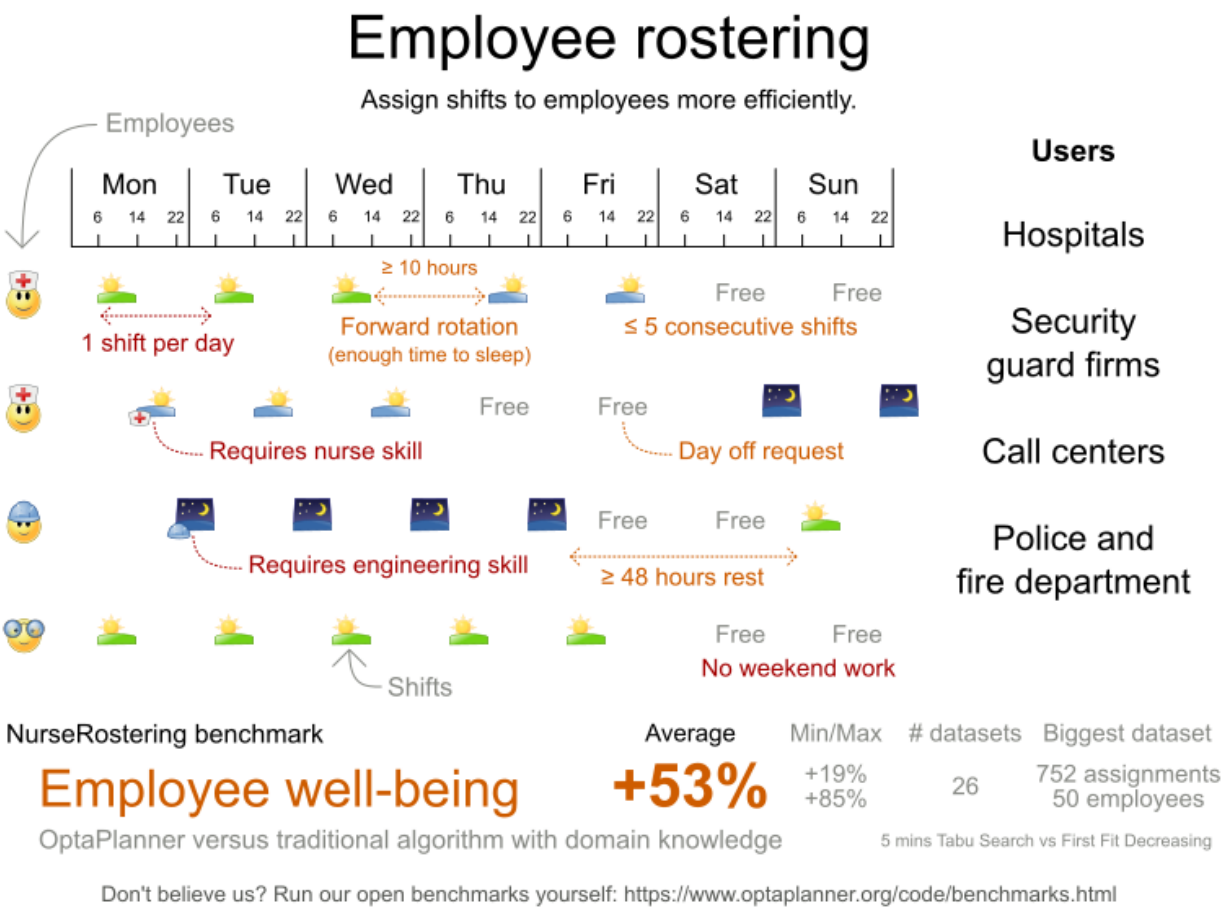
- 契約上の義務。この業界では、頻繁に契約上の義務に違反するため、ハード制約ではなく、ソフト制約として定義することに決定しました。
  - 割り当ての下限および上限: 各従業員は、(それぞれの契約に合わせて) x より多く、y よりも少ないシフト数を勤務する必要がある。
  - 連続勤務日数の下限および上限: 各従業員は、(それぞれの契約に合わせて) 連続で x 日から y 日間、勤務する必要がある。
  - 連続公休日数の下限および上限: 各従業員は、(それぞれの契約に合わせて) 連続で x 日から y 日間、休む必要がある。
  - 週末に連続勤務する回数の下限および上限: 各従業員は、(それぞれの契約に合わせて) 連続で x 回から y 回、週末勤務する必要がある。
  - 週末の勤務有無を同じにする: 各従業員は、週末の両日を勤務する、または休む必要がある。
  - 週末のシフトタイプを同じにする: 各従業員で、同じ週末のシフトタイプは、同じにする必要がある。

- 不要なパターン - 行に不要なシフトタイプの組み合わせ。例: 遅いシフト、早いシフト、遅いシフト。

- 従業員の希望:
  - 勤務日のリクエスト: 従業員は、特定の勤務希望日を申請できる。
  - 公休日のリクエスト: 従業員は、特定の公休希望日を申請できる。
  - 勤務するシフトのリクエスト: 従業員は特定のシフトへの割り当てを希望できる。
  - 勤務しないシフトのリクエスト: 従業員は特定のシフトに割り当てられないように希望できる。
- 他のスキル: スキルに割り当てられた従業員は、そのシフトで必要な全スキルに堪能である必要がある。

この問題は「International Nurse Rostering Competition 2010」で定義されています。

図4.12 価値提案



問題の規模

以下のように、データセットの種類は 3 つあります。

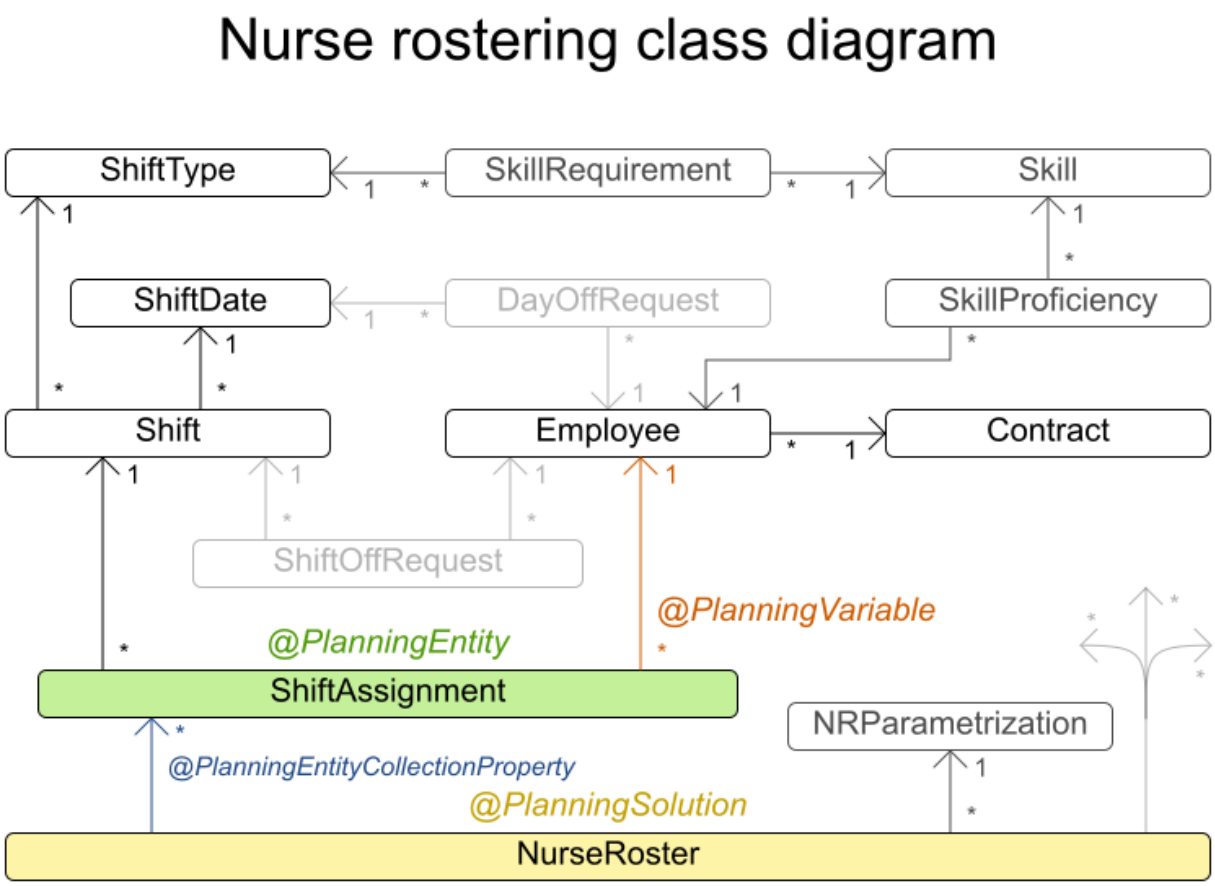
- **sprint:** 数秒で問題を解決する必要があります。
- **medium:** 数分で問題を解決する必要があります。
- **long:** 数時間で問題を解決する必要があります。

```
toy1          has 1 skills, 3 shiftTypes, 2 patterns, 1 contracts,  6 employees,  7 shiftDates,  35
shiftAssignments and   0 requests with a search space of   10^27.
toy2          has 1 skills, 3 shiftTypes, 3 patterns, 2 contracts, 20 employees, 28 shiftDates, 180
shiftAssignments and 140 requests with a search space of  10^234.

sprint01      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint02      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint03      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint04      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint05      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint06      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint07      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint08      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint09      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint10      has 1 skills, 4 shiftTypes, 3 patterns, 4 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_hint01 has 1 skills, 4 shiftTypes, 8 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_hint02 has 1 skills, 4 shiftTypes, 0 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_hint03 has 1 skills, 4 shiftTypes, 8 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_late01 has 1 skills, 4 shiftTypes, 8 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_late02 has 1 skills, 3 shiftTypes, 4 patterns, 3 contracts, 10 employees, 28 shiftDates, 144
shiftAssignments and 139 requests with a search space of  10^144.
sprint_late03 has 1 skills, 4 shiftTypes, 8 patterns, 3 contracts, 10 employees, 28 shiftDates, 160
shiftAssignments and 150 requests with a search space of  10^160.
sprint_late04 has 1 skills, 4 shiftTypes, 8 patterns, 3 contracts, 10 employees, 28 shiftDates, 160
shiftAssignments and 150 requests with a search space of  10^160.
sprint_late05 has 1 skills, 4 shiftTypes, 8 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_late06 has 1 skills, 4 shiftTypes, 0 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_late07 has 1 skills, 4 shiftTypes, 0 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.
sprint_late08 has 1 skills, 4 shiftTypes, 0 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and   0 requests with a search space of  10^152.
sprint_late09 has 1 skills, 4 shiftTypes, 0 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and   0 requests with a search space of  10^152.
sprint_late10 has 1 skills, 4 shiftTypes, 0 patterns, 3 contracts, 10 employees, 28 shiftDates, 152
shiftAssignments and 150 requests with a search space of  10^152.

medium01      has 1 skills, 4 shiftTypes, 0 patterns, 4 contracts, 31 employees, 28 shiftDates, 608
shiftAssignments and 403 requests with a search space of  10^906.
medium02      has 1 skills, 4 shiftTypes, 0 patterns, 4 contracts, 31 employees, 28 shiftDates, 608
shiftAssignments and 403 requests with a search space of  10^906.
```

medium03 has 1 skills, 4 shiftTypes, 0 patterns, 4 contracts, 31 employees, 28 shiftDates, 608 shiftAssignments and 403 requests with a search space of 10^906.
medium04 has 1 skills, 4 shiftTypes, 0 patterns, 4 contracts, 31 employees, 28 shiftDates, 608 shiftAssignments and 403 requests with a search space of 10^906.
medium05 has 1 skills, 4 shiftTypes, 0 patterns, 4 contracts, 31 employees, 28 shiftDates, 608 shiftAssignments and 403 requests with a search space of 10^906.
medium_hint01 has 1 skills, 4 shiftTypes, 7 patterns, 4 contracts, 30 employees, 28 shiftDates, 428 shiftAssignments and 390 requests with a search space of 10^632.
medium_hint02 has 1 skills, 4 shiftTypes, 7 patterns, 3 contracts, 30 employees, 28 shiftDates, 428 shiftAssignments and 390 requests with a search space of 10^632.
medium_hint03 has 1 skills, 4 shiftTypes, 7 patterns, 4 contracts, 30 employees, 28 shiftDates, 428 shiftAssignments and 390 requests with a search space of 10^632.
medium_late01 has 1 skills, 4 shiftTypes, 7 patterns, 4 contracts, 30 employees, 28 shiftDates, 424 shiftAssignments and 390 requests with a search space of 10^626.
medium_late02 has 1 skills, 4 shiftTypes, 7 patterns, 3 contracts, 30 employees, 28 shiftDates, 428 shiftAssignments and 390 requests with a search space of 10^632.
medium_late03 has 1 skills, 4 shiftTypes, 0 patterns, 4 contracts, 30 employees, 28 shiftDates, 428 shiftAssignments and 390 requests with a search space of 10^632.
medium_late04 has 1 skills, 4 shiftTypes, 7 patterns, 3 contracts, 30 employees, 28 shiftDates, 416 shiftAssignments and 390 requests with a search space of 10^614.
medium_late05 has 2 skills, 5 shiftTypes, 7 patterns, 4 contracts, 30 employees, 28 shiftDates, 452 shiftAssignments and 390 requests with a search space of 10^667.

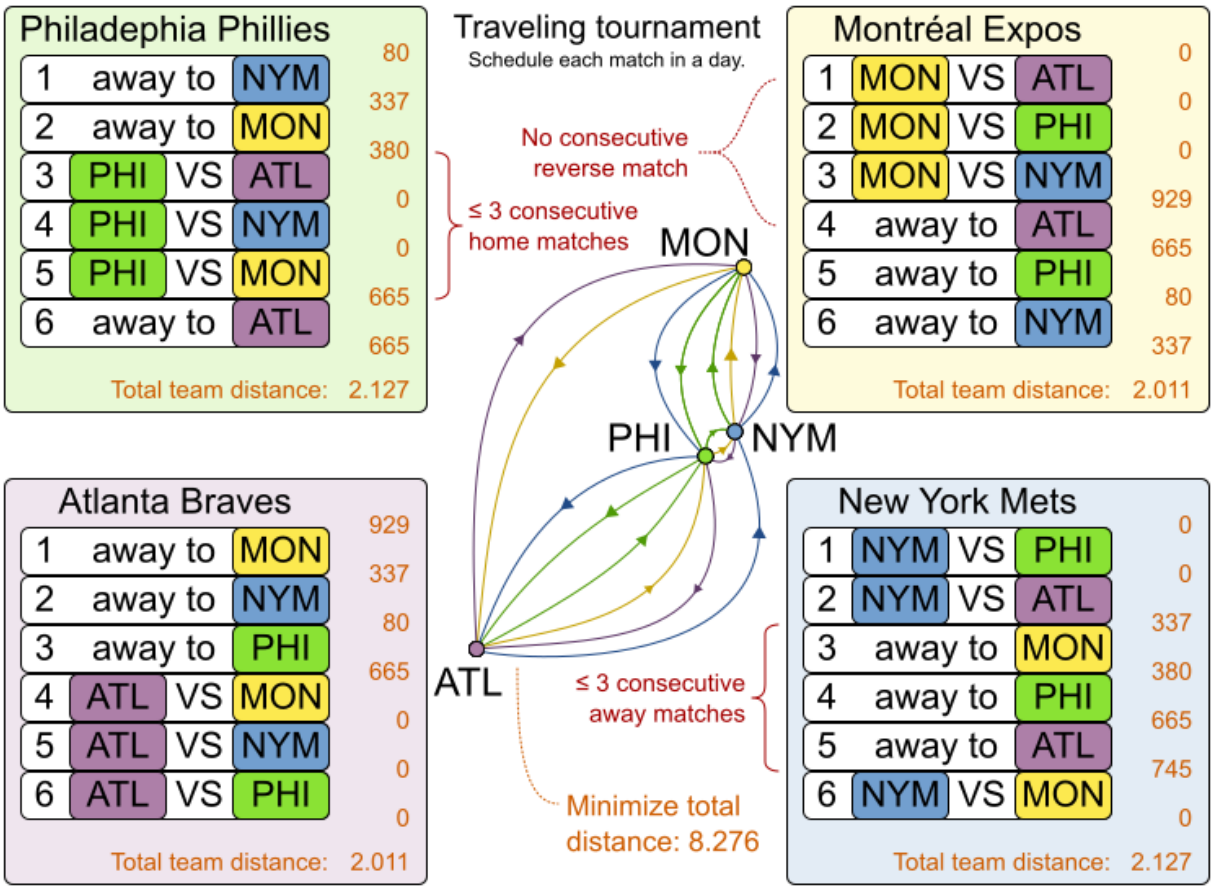
long01 has 2 skills, 5 shiftTypes, 3 patterns, 3 contracts, 49 employees, 28 shiftDates, 740 shiftAssignments and 735 requests with a search space of 10^1250.
long02 has 2 skills, 5 shiftTypes, 3 patterns, 3 contracts, 49 employees, 28 shiftDates, 740 shiftAssignments and 735 requests with a search space of 10^1250.
long03 has 2 skills, 5 shiftTypes, 3 patterns, 3 contracts, 49 employees, 28 shiftDates, 740 shiftAssignments and 735 requests with a search space of 10^1250.
long04 has 2 skills, 5 shiftTypes, 3 patterns, 3 contracts, 49 employees, 28 shiftDates, 740 shiftAssignments and 735 requests with a search space of 10^1250.
long05 has 2 skills, 5 shiftTypes, 3 patterns, 3 contracts, 49 employees, 28 shiftDates, 740 shiftAssignments and 735 requests with a search space of 10^1250.
long_hint01 has 2 skills, 5 shiftTypes, 9 patterns, 3 contracts, 50 employees, 28 shiftDates, 740 shiftAssignments and 0 requests with a search space of 10^1257.
long_hint02 has 2 skills, 5 shiftTypes, 7 patterns, 3 contracts, 50 employees, 28 shiftDates, 740 shiftAssignments and 0 requests with a search space of 10^1257.
long_hint03 has 2 skills, 5 shiftTypes, 7 patterns, 3 contracts, 50 employees, 28 shiftDates, 740 shiftAssignments and 0 requests with a search space of 10^1257.
long_late01 has 2 skills, 5 shiftTypes, 9 patterns, 3 contracts, 50 employees, 28 shiftDates, 752 shiftAssignments and 0 requests with a search space of 10^1277.
long_late02 has 2 skills, 5 shiftTypes, 9 patterns, 4 contracts, 50 employees, 28 shiftDates, 752 shiftAssignments and 0 requests with a search space of 10^1277.
long_late03 has 2 skills, 5 shiftTypes, 9 patterns, 3 contracts, 50 employees, 28 shiftDates, 752 shiftAssignments and 0 requests with a search space of 10^1277.
long_late04 has 2 skills, 5 shiftTypes, 9 patterns, 4 contracts, 50 employees, 28 shiftDates, 752 shiftAssignments and 0 requests with a search space of 10^1277.
long_late05 has 2 skills, 5 shiftTypes, 9 patterns, 3 contracts, 50 employees, 28 shiftDates, 740 shiftAssignments and 0 requests with a search space of 10^1257.

図4.13 ドメインモデル



4.16. 巡回トーナメント問題 (TTP)

n チーム間の試合をスケジュールします。



ハード制約:

- 各チームは、他のチームとそれぞれ 2 回 (ホームとアウェイ) 試合をする。
- 各チームは、各時間枠に 1 試合だけ行う。
- 3 回連続で、ホームまたはアウェイでの試合はできない。
- 繰り返しなし: 同じ対戦相手と 2 回連続で対戦できない。

ソフト制約:

- 全チームが移動する合計距離を最小限に抑える。

この問題は Michael Trick の Web サイト (世界記録が含まれます) で定義されています。

問題の規模

```
1-nl04    has  6 days,  4 teams and   12 matches with a search space of    10^5.
1-nl06    has 10 days,  6 teams and   30 matches with a search space of   10^19.
1-nl08    has 14 days,  8 teams and   56 matches with a search space of   10^43.
1-nl10    has 18 days, 10 teams and   90 matches with a search space of   10^79.
1-nl12    has 22 days, 12 teams and  132 matches with a search space of  10^126.
1-nl14    has 26 days, 14 teams and  182 matches with a search space of  10^186.
1-nl16    has 30 days, 16 teams and  240 matches with a search space of  10^259.
2-bra24    has 46 days, 24 teams and  552 matches with a search space of  10^692.
3-nfl16   has 30 days, 16 teams and  240 matches with a search space of  10^259.
3-nfl18   has 34 days, 18 teams and  306 matches with a search space of  10^346.
```

```
3-nfl20    has 38 days, 20 teams and  380 matches with a search space of  10^447.
3-nfl22    has 42 days, 22 teams and  462 matches with a search space of  10^562.
3-nfl24    has 46 days, 24 teams and  552 matches with a search space of  10^692.
3-nfl26    has 50 days, 26 teams and  650 matches with a search space of  10^838.
3-nfl28    has 54 days, 28 teams and  756 matches with a search space of  10^999.
3-nfl30    has 58 days, 30 teams and  870 matches with a search space of 10^1175.
3-nfl32    has 62 days, 32 teams and  992 matches with a search space of 10^1367.
4-super04  has  6 days,  4 teams and   12 matches with a search space of    10^5.
4-super06  has 10 days,  6 teams and   30 matches with a search space of   10^19.
4-super08  has 14 days,  8 teams and   56 matches with a search space of   10^43.
4-super10  has 18 days, 10 teams and   90 matches with a search space of   10^79.
4-super12  has 22 days, 12 teams and  132 matches with a search space of  10^126.
4-super14  has 26 days, 14 teams and  182 matches with a search space of  10^186.
5-galaxy04 has  6 days,  4 teams and   12 matches with a search space of    10^5.
5-galaxy06 has 10 days,  6 teams and   30 matches with a search space of   10^19.
5-galaxy08 has 14 days,  8 teams and   56 matches with a search space of   10^43.
5-galaxy10 has 18 days, 10 teams and   90 matches with a search space of   10^79.
5-galaxy12 has 22 days, 12 teams and  132 matches with a search space of  10^126.
5-galaxy14 has 26 days, 14 teams and  182 matches with a search space of  10^186.
5-galaxy16 has 30 days, 16 teams and  240 matches with a search space of  10^259.
5-galaxy18 has 34 days, 18 teams and  306 matches with a search space of  10^346.
5-galaxy20 has 38 days, 20 teams and  380 matches with a search space of  10^447.
5-galaxy22 has 42 days, 22 teams and  462 matches with a search space of  10^562.
5-galaxy24 has 46 days, 24 teams and  552 matches with a search space of  10^692.
5-galaxy26 has 50 days, 26 teams and  650 matches with a search space of  10^838.
5-galaxy28 has 54 days, 28 teams and  756 matches with a search space of  10^999.
5-galaxy30 has 58 days, 30 teams and  870 matches with a search space of 10^1175.
5-galaxy32 has 62 days, 32 teams and  992 matches with a search space of 10^1367.
5-galaxy34 has 66 days, 34 teams and 1122 matches with a search space of 10^1576.
5-galaxy36 has 70 days, 36 teams and 1260 matches with a search space of 10^1801.
5-galaxy38 has 74 days, 38 teams and 1406 matches with a search space of 10^2042.
5-galaxy40 has 78 days, 40 teams and 1560 matches with a search space of 10^2301.
```

4.17. コストを抑えるスケジュール

全タスクを時間内にスケジュールし、機械の電気代を最小限に抑えます。電気代は時間によって異なります。これは、ジョブショップスケジューリング の応用です。

ハード制約:

- 開始時間の制限: 各タスクは、最早と最遅の開始時間の制限内に、開始する必要がある。
- 最大容量: マシンに割り当てる各リソースはこの量を超えてはいけない。
- 開始および終了: 各機械は、タスクが割り当てられている間は稼働している必要がある。次のタスクまでの間、起動および終了コストを避けるため、機械をアイドルにすることができる。

中程度の制約:

- 電気代: 全スケジュールの合計電気代を最小限に抑える。
  - 機械の電気代: 稼働中またはアイドル中の機械はそれぞれ、電気を消費し、電気代が発生する (金額は使用時の電気代によって異なる)。
  - タスクの電気代: 各タスクも電気を消費し、電気代が発生する (金額は使用時の電気代によって異なる)。

- 機械の起動および終了コスト: 機械を起動または終了するたびに、追加のコストが発生する。

ソフト制約 (問題に元々設定されている定義に追加):

- 早く開始: なるべく早めにタスクを開始するようにする。

この問題は、「ICON challenge」で定義されています。

問題の規模

```
sample01   has 3 resources,   2 machines, 288 periods and   25 tasks with a search space of
10^53.
sample02   has 3 resources,   2 machines, 288 periods and   50 tasks with a search space of
10^114.
sample03   has 3 resources,   2 machines, 288 periods and  100 tasks with a search space of
10^226.
sample04   has 3 resources,   5 machines, 288 periods and  100 tasks with a search space of
10^266.
sample05   has 3 resources,   2 machines, 288 periods and  250 tasks with a search space of
10^584.
sample06   has 3 resources,   5 machines, 288 periods and  250 tasks with a search space of
10^673.
sample07   has 3 resources,   2 machines, 288 periods and 1000 tasks with a search space of
10^2388.
sample08   has 3 resources,   5 machines, 288 periods and 1000 tasks with a search space of
10^2748.
sample09   has 4 resources,  20 machines, 288 periods and 2000 tasks with a search space of
10^6668.
instance00 has 1 resources,  10 machines, 288 periods and  200 tasks with a search space of
10^595.
instance01 has 1 resources,  10 machines, 288 periods and  200 tasks with a search space of
10^599.
instance02 has 1 resources,  10 machines, 288 periods and  200 tasks with a search space of
10^599.
instance03 has 1 resources,  10 machines, 288 periods and  200 tasks with a search space of
10^591.
instance04 has 1 resources,  10 machines, 288 periods and  200 tasks with a search space of
10^590.
instance05 has 2 resources,  25 machines, 288 periods and  200 tasks with a search space of
10^667.
instance06 has 2 resources,  25 machines, 288 periods and  200 tasks with a search space of
10^660.
instance07 has 2 resources,  25 machines, 288 periods and  200 tasks with a search space of
10^662.
instance08 has 2 resources,  25 machines, 288 periods and  200 tasks with a search space of
10^651.
instance09 has 2 resources,  25 machines, 288 periods and  200 tasks with a search space of
10^659.
instance10 has 2 resources,  20 machines, 288 periods and  500 tasks with a search space of
10^1657.
instance11 has 2 resources,  20 machines, 288 periods and  500 tasks with a search space of
10^1644.
instance12 has 2 resources,  20 machines, 288 periods and  500 tasks with a search space of
10^1637.
instance13 has 2 resources,  20 machines, 288 periods and  500 tasks with a search space of
```

```
10^1659.
instance14 has 2 resources,  20 machines, 288 periods and  500 tasks with a search space of
10^1643.
instance15 has 3 resources,  40 machines, 288 periods and  500 tasks with a search space of
10^1782.
instance16 has 3 resources,  40 machines, 288 periods and  500 tasks with a search space of
10^1778.
instance17 has 3 resources,  40 machines, 288 periods and  500 tasks with a search space of
10^1764.
instance18 has 3 resources,  40 machines, 288 periods and  500 tasks with a search space of
10^1769.
instance19 has 3 resources,  40 machines, 288 periods and  500 tasks with a search space of
10^1778.
instance20 has 3 resources,  50 machines, 288 periods and 1000 tasks with a search space of
10^3689.
instance21 has 3 resources,  50 machines, 288 periods and 1000 tasks with a search space of
10^3678.
instance22 has 3 resources,  50 machines, 288 periods and 1000 tasks with a search space of
10^3706.
instance23 has 3 resources,  50 machines, 288 periods and 1000 tasks with a search space of
10^3676.
instance24 has 3 resources,  50 machines, 288 periods and 1000 tasks with a search space of
10^3681.
instance25 has 3 resources,  60 machines, 288 periods and 1000 tasks with a search space of
10^3774.
instance26 has 3 resources,  60 machines, 288 periods and 1000 tasks with a search space of
10^3737.
instance27 has 3 resources,  60 machines, 288 periods and 1000 tasks with a search space of
10^3744.
instance28 has 3 resources,  60 machines, 288 periods and 1000 tasks with a search space of
10^3731.
instance29 has 3 resources,  60 machines, 288 periods and 1000 tasks with a search space of
10^3746.
instance30 has 4 resources,  70 machines, 288 periods and 2000 tasks with a search space of
10^7718.
instance31 has 4 resources,  70 machines, 288 periods and 2000 tasks with a search space of
10^7740.
instance32 has 4 resources,  70 machines, 288 periods and 2000 tasks with a search space of
10^7686.
instance33 has 4 resources,  70 machines, 288 periods and 2000 tasks with a search space of
10^7672.
instance34 has 4 resources,  70 machines, 288 periods and 2000 tasks with a search space of
10^7695.
instance35 has 4 resources,  80 machines, 288 periods and 2000 tasks with a search space of
10^7807.
instance36 has 4 resources,  80 machines, 288 periods and 2000 tasks with a search space of
10^7814.
instance37 has 4 resources,  80 machines, 288 periods and 2000 tasks with a search space of
10^7764.
instance38 has 4 resources,  80 machines, 288 periods and 2000 tasks with a search space of
10^7736.
instance39 has 4 resources,  80 machines, 288 periods and 2000 tasks with a search space of
10^7783.
instance40 has 4 resources,  90 machines, 288 periods and 4000 tasks with a search space of
10^15976.
instance41 has 4 resources,  90 machines, 288 periods and 4000 tasks with a search space of
```

> 10^15935.
> instance42 has 4 resources, 90 machines, 288 periods and 4000 tasks with a search space of 10^15887.
> instance43 has 4 resources, 90 machines, 288 periods and 4000 tasks with a search space of 10^15896.
> instance44 has 4 resources, 90 machines, 288 periods and 4000 tasks with a search space of 10^15885.
> instance45 has 4 resources, 100 machines, 288 periods and 5000 tasks with a search space of 10^20173.
> instance46 has 4 resources, 100 machines, 288 periods and 5000 tasks with a search space of 10^20132.
> instance47 has 4 resources, 100 machines, 288 periods and 5000 tasks with a search space of 10^20126.
> instance48 has 4 resources, 100 machines, 288 periods and 5000 tasks with a search space of 10^20110.
> instance49 has 4 resources, 100 machines, 288 periods and 5000 tasks with a search space of 10^20078.

4.18. 投資資産クラスの割り当て (ポートフォリオの最適化)

各資産クラスに投資する相対数を決定します。

ハード制約:

- リスクの最大値: 標準偏差合計は、標準偏差の最大値を超えてはならない。
  - 標準偏差合計の計算は、Markowitz Portfolio Theoryを適用した、資産クラスの相対関係を考慮する必要がある。
- 地域の最大値: 地域ごとに数量の最大値がある。
- セクターの最大値: 各セクターに数量の最大値がある。

ソフト制約:

- 期待収益を最大化する。

問題の規模

> de_smet_1 has 1 regions, 3 sectors and 11 asset classes with a search space of 10^4.
> irrinki_1 has 2 regions, 3 sectors and 6 asset classes with a search space of 10^3.

サイズが大きいデータセットは作成/検証されていませんが、問題はないはずです。データに関する適切な情報源として、このアセット相関の Web サイト を参照してください。

4.19. 会議スケジュール

各会議を時間帯と部屋に割り当てていきます。時間帯は重複させることができます。***.xlsx** ファイルに対する読み取りや書き込みが可能で、このファイルは LibreOffice または Microsof Excel で編集できます。

ハード制約:

- 時間帯の会議タイプ: 会議のタイプは、時間帯の会議タイプと一致する必要がある。

- 部屋が使用中の時間帯: その会議の時間帯に、会議用の部屋が利用できなければならない。
- 部屋の競合: 2 つの会議が、同じ時間に同じ会議室を使用することはできない。
- 講演者が空いていない時間帯: 講演者は必ず、会議の時間帯に空いていなければならない。
- 講演者の競合: 同じ時間帯の 2 つの会議に同じ講演者を割り当てることができない。
- 汎用の時間帯および部屋のタグ
  - 講演者が要求する時間帯タグ: 講演者に、必須時間帯タグが付けられている場合、講演者の会議はそのタグが付いている時間に割り当てる必要がある。
  - 講演者の禁止時間帯タグ: 公演者に、禁止時間帯タグが割り当てられている場合は、そのタグの付いた時間帯に講演者の会議をどれも割り当てることができない。
  - 会議を設定する必要のある時間帯タグ: 会議に必須時間帯タグが付いている場合は、そのタグの付いた時間帯に割り当てる必要がある。
  - 会議の禁止時間帯タグ: 会議に、禁止時間帯タグが割り当てられている場合は、そのタグの付いた時間帯にその会議を割り当てることができない。
  - 講演者が要求する部屋のタグ: 講演者に、必須の部屋タグが付けられている場合、講演者の会議はそのタグが付いている部屋に割り当てる必要がある。
  - 講演者が禁止する部屋のタグ: 講演者に、禁止部屋のタグが付けられている場合、講演者の会議はそのタグが付いている部屋に割り当てことができない。
  - 会議を設定する必要のある部屋タグ: 会議に必須部屋タグが付いている場合は、そのタグの付いた部屋に割り当てる必要がある。
  - 会議の禁止部屋タグ: 会議に、禁止部屋タグが割り当てられている場合は、そのタグの付いた部屋にその会議を割り当てることができない。
- 他の会議と同じ時間帯に設定しないタグ: このタグが付いている会議は、同じ時間帯に重複してスケジュールしてはいけない。
- 受講条件が付いた会議: 受講条件が付いた会議をすべて完了してからでないと対象の会議をスケジュールしてはいけない。

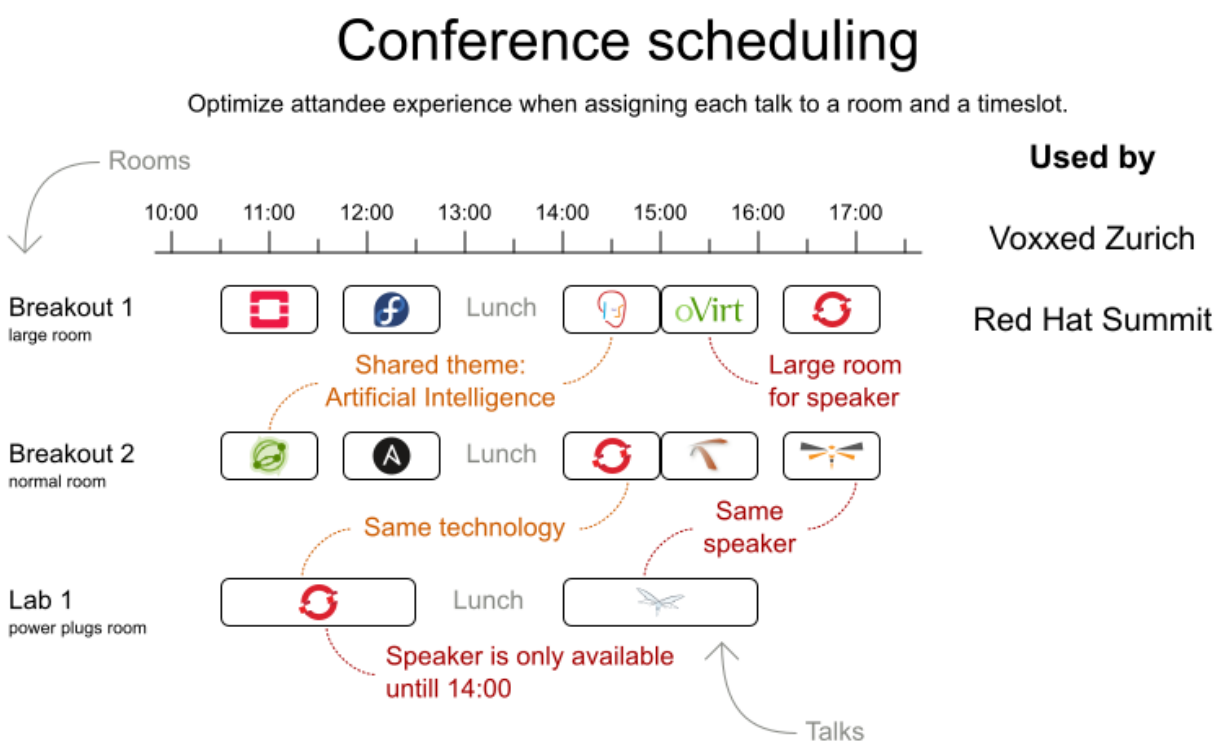
ソフト制約:

- テーマの追跡競合: 同じ時間帯で、同じテーマのタグが付いた会議の数を最小限に抑える。
- セクターの競合: 同じ時間帯で同じセクタータグの付いた会議の数を最小限に抑える。
- コンテンツの受講者レベルのフロー違反: すべてのコンテンツタグに対して、上級者用の会議の前に入門レベルの会議をスケジュールする。
- 受講者レベルの多様性: すべての時間帯において、異なる受講者レベルの会議数を最大限に増やす。
- 言語の多様性: すべての時間帯において、異なる言語の会議数を最大限を増やす。
- 汎用の時間帯および部屋のタグ
  - 講演者が希望する時間帯タグ: 講演者に、希望の時間帯タグが付けられている場合、講演者の会議はそのタグが付いている時間に割り当てるようにする。

- 講演者が希望しない時間帯タグ: 講演者に、希望しない時間帯タグが付けられている場合、その講演者の会議はそのタグが付いている時間に割り当てないようにする。
- 会議の希望の時間帯タグ: 会議に希望の時間帯タグが付いている場合は、そのタグの付いた時間帯に割り当てるようにする。
- 会議の設定を希望しない時間帯タグ: 会議に、希望しない時間帯タグが付いている場合は、そのタグの付いた時間帯に割り当てないようにする。
- 講演者が希望する部屋のタグ: 講演者に、希望の部屋タグが付けられている場合、講演者の会議はそのタグが付いている部屋に割り当てるようにする。
- 講演者が希望しない部屋のタグ: 講演者に、希望しない部屋タグが付けられている場合、講演者の会議はそのタグが付いている部屋に割り当てないようにする。
- 会議を希望の部屋タグ: 会議に希望の部屋タグが付いている場合は、そのタグの付いた部屋に割り当てるようにする。
- 会議での使用を希望しない部屋タグ: 会議に、希望しない部屋タグが付いている場合、そのタグの付いた部屋に割り当てないようにする。

- 同じ日の会議: 同じテーマタグまたはコンテンツタグが付いた会議は必ず、必要最小限に日数 (理想としては同じ日) にスケジュールするようにする。

図4.14 価値提案



問題の規模

> 18talks-6timeslots-5rooms has 18 talks, 6 timeslots and 5 rooms with a search space of 10^26.
> 36talks-12timeslots-5rooms has 36 talks, 12 timeslots and 5 rooms with a search space of 10^64.
> 72talks-12timeslots-10rooms has 72 talks, 12 timeslots and 10 rooms with a search space of 10^149.
> 108talks-18timeslots-10rooms has 108 talks, 18 timeslots and 10 rooms with a search space of 10^243.
> 216talks-18timeslots-20rooms has 216 talks, 18 timeslots and 20 rooms with a search space of 10^552.

4.20. ロックツアー

次のショーへの移動はロックバスを使用し、空いている日のみショーをスケジュールする。

ハード制約:

- 必要とされるショーをすべてスケジュールする。
- できるだけ多くのショーをスケジュールする。

中程度の制約:

- 収益の機会を最大化する。
- 運転時間を最小限に抑える。
- できるだけ早く到着する。

ソフト制約:

- 長時間の運転は避ける。

問題の規模

> 47shows has 47 shows with a search space of 10^59.

4.21. 航空機乗組員のスケジューリング

パイロットと客室乗務員にフライトを割り当てます。

ハード制約:

- 必須スキル: フライトの割り当てにはそれぞれ、必要とされるスキルがあります。たとえば、フライト AB0001 ではパイロット 2 名と、客室乗務員 3 名が必要です。
- フライトの競合: 各従業員は同じ時間に出勤できるフライトは 1 つだけにする。
- 2 つのフライト間での移動: 2 つのフライトの間で、従業員は到着先の空港と、出発元の空港に移動できる必要がある。たとえば、アンは 10 時にブリュッセルに到着し、15 時にアムステルダムを出発するなどです。
- 従業員の勤務できない日: 従業員はフライトの当日は空いていなければならない。たとえば、アンは 2 月 1 日に休暇を取っているなど。

ソフト制約:

- 最初の仕事が自宅から出発する。
- 最後の仕事が自宅に到着する。
- 総フライト時間を従業員別に平均的に分散する。

問題の規模

> 175flights-7days-Europe has 2 skills, 50 airports, 150 employees, 175 flights and 875 flight assignments with a search space of 10^1904.
> 700flights-28days-Europe has 2 skills, 50 airports, 150 employees, 700 flights and 3500 flight assignments with a search space of 10^7616.
> 875flights-7days-Europe has 2 skills, 50 airports, 750 employees, 875 flights and 4375 flight assignments with a search space of 10^12578.
> 175flights-7days-US has 2 skills, 48 airports, 150 employees, 175 flights and 875 flight assignments with a search space of 10^1904.

付録A バージョン情報

本書の最終更新日： 2021 年 7 月 19 日（月）