第4章 プログラミング支援機能が充実！クラウド基盤「Fixstars Amplify」でトライ③

チェーン店の人員配置最適化に挑戦

広田 望

第2章と第3章で，Fixstars Amplifyを使ったプログラミングの基本を一通り解説しました．最後に実際の開発事例にもあった参考問題に挑戦してみましょう．飲食チェーン店での勤務地を決める最適化問題です．このプログラミングでは，一度作ったソフトウェアを改修する工程も体験します．

● 数式を考えるのは計算以上に難しい

挑戦するのは，飲食チェーンなどでどの店舗で働いてもらうか従業員を割り振るものです．従業員には勤務地の希望があり，もちろん勤務できない店舗もあります．店舗運営に必要な人員を十分に配置しつつ，できるだけ従業員の希望が多くかなうように割り当てパターンを考えていきます．こうした実問題に取り組むとき，難しいのが「店舗運営に必要な人員」という条件です．

組み合わせ最適化問題として実問題に取り組むとき，どうすれば解きたい問題を数式にできるかが最大のハードルかもしれません．言葉では簡単に説明できても，そこからどんな数式を定義するべきなのかを判断するのは案外難しいものです．例えばこの問題では，1人の従業員は同時に1店舗にしか勤務できないという制約を数式で設定しなければなりません．あまりに基本的な条件なので，おそらく紙とペンで勤務地割り当てを考える場合には意識しない条件です．

定式化した数式が正しく解きたい問題を表せているかどうかは，実際に計算してみないと判断できないことも多くあります．実際の開発プロセスでは，サンプル・データを使いながらソフトウェアの改修と実行を繰り返し，問題の条件を確認していきます．実行結果を見て追加で考慮すべき条件が見つかったら，制約条件や目的関数などを見直すといった手順です．

第1弾のコード全容は**リスト1**に示します．

ステップ1：サンプル・データの生成と決定変数の実装

● 条件設定（リスト1の①）

最初は計算に使うサンプル・データの準備です．表形式でのデータ表示を使いたいので，データ操作パッケージpandasを使っています．Jupyter NotebookなどのPython実行環境でデータ表示をするのに便利です．

勤務地は博多店と天神店，従業員は5人いることにします．天神店の必要従業員数は2人，博多店は3人です．従業員の勤務地希望は，勤務希望（2），希望はしないが勤務可（1），勤務不可（0）の3段階で集計します．**リスト1**の①の部分が，これらのサンプル・データを生成しているパートです．コメントで「dictの作成」としているのは，`tenjin`や`hakata`といった店舗名での表記を，対応する店舗番号と相互変換するための辞書型変数です．

● 組み合わせを表現できる決定変数を考える（リスト1の②）

次は決定変数を考えます．どの従業員がどの店舗に勤務するかを決めるのが計算の目的なので，その組み合わせを表現できる決定変数を考えます．例では2店舗のみですが，より多くの店舗にも対応しやすい実装方法として，従業員と店舗の組み合わせごとに1つのバイナリ変数を用意することにします．

従業員iが店舗lに勤務するなら，バイナリ変数$L_{i,\,l}$が1になるという定義です．決定変数L（`location_variables`）は従業員ID（`num_workers`）と店舗番号（`num_locations`）で操作しやすいよう，2重配列で用意することにします．

用意した決定変数のうち，事前に答えが明らかなものは値を固定しておきます．勤務が不可能な店舗には割り当てされないように，事前に該当する決定変数の値を0にしておきます．勤務不可は希望度（`worker_req`）の値が0なので，該当する決定変数をバイナリ変数型の0に上書きします．

ステップ2：目的関数の実装

● 従業員充足率（リスト1の③）

次に目的関数をどのように定義するか考えます．今回の目的はできるだけ多くの従業員の希望がかなう割