Intelligence Programming I
Final Exercise

Submitted on 2014/08/07 • CSb 24115113 Masayuki Hayashi (林 政行)

Theme (テーマ)

Tasks to Schedule (スケジュール自動生成)

Why I chose this theme (テーマ選択理由)

I chose 'Tasks to Schedule' as the theme. There are mainly two reasons for choosing this theme: my long-cherished program, and what I have learned from the past exercise.

I often have had an feeling that, if I could completely rely on a program for making my own schedule and managing it, that would be useful and could reduce such mistakes as wasting time without recognizing exactly what I need to do at a certain time or completely forgetteing the think I actually need to do. Therefore, I have wanted such, in a sense, 'artificial intelligence program.'

Having such feeling, the things I have learned through the exercises of this class made me have an idea that I may be able to make a program that generates my own schedule from the tasks that I have. Therefore, I chose this subject as my theme this time.

このテーマ「スケジュール自動生成」を選んだ理由として、自分のスケジュール管理を自動で行ってくれるプログラムがかねてから欲しかったこと、ならびに、今回の演習の応用課題を通して、制約充足問題を Prolog で解く方法が学べ、それを今回の課題の解決で使えると考えたからである。

こなすべきタスクから最適なスケジューリングを作成し、それを管理してくれるようなプログラムがあれば、今やるべきことを正しく認識し、期日に間に合わなかったり、完全にすべきことを忘れていた、といったミスを減らすことができるだろうという考えもあり、そのようなある種の AI プログラムが欲しかった。

このような考えを持ち、本演習で学んだことを踏まえ、もしかしたらスケジュール自動生成を行うプログラムが作れるかもしれないと思い、今回このテーマを選択した。

プログラム内容の概要

今回作成したプログラムでは、タスクからスケジュールを作成する処理を、重み付き制約充足問題 (VCSP) を解くことで実現することを試みた。人がスケジュールを作成する際に、タスクの実行順序や期日を考慮してスケジュール日時を決める、という方法をとると考えられるが、これらのスケジュール作成の際に考慮すべき内容は、制約としてとらえることができ、各制約に異なる重みを持たせた VCSP を解くことで、制約の優先度を考慮したスケジュールが作れるようになると考え、このような方法をとった。

今回の問題を VCSP に定式化するにあたって、スケジュール期間をある範囲 (N_{day} 日間,N_{time} 時間) に絞った。その上で各日のスケジュール対象の時間、およびタスクを 10 分ごとに分割し、変数 $x_{ij} \in X$ を i 日目の j 時間目、値域 D を分割したタスク、制約の集合 C を前述したスケジュール作成の際に考慮すべき条件とし、適当な評価構造 S、評価関数 ϕ を通常の加算として、VCSP における今

回のスケジューリングの問題 VP を $VP = (X, D, C, S, \phi)$ という構造で表現した。

制約充足問題を解く方法としては、各変数に順番に、制約を満たす値を割り当て、解を見つける方法、各変数間の制約、アークに注目し、アーク無矛盾な値の割り当てを見つけることで解を見つける方法、といった完全アルゴリズムがあるが、今回は計算量を減らすことを考え、本演習の応用課題で用いた Min-Conflicts, ならびに局所探索をすることで解を探す Stochastic Hill Climbing を組み合わせた、近似アルゴリズムを使って解を得る手法を用いた。

工夫したところ

工夫した点としては、再スケジューリング時に前回のスケジューリング結果から大幅な変更が内容にする制約を加える点、できるだけ局所解に至らないように、かつ解が収束するように、Simulated Annealing を踏まえたランダムな値選択を行う方法をとった点である。

前者は意図した通り、再スケジューリング時に前回のスケジューリング結果とはそれほど変わらないようにする効果が得られたが、後者は、考慮が足りなかった部分もあり、制約が少ない場合に、ランダム値の選択を無駄に行うために、解の収束に時間が掛かってしまう結果を招いた。

問題点

今回作成したプログラムでは、制約の重みを最後まで適切に設定することができず、制約が充足している解においても評価値が 0 にならず、無限に解の探索を続けてしまう問題を持っている。この問題の表面上の解決を図り、Simulated Annealing を用いて、解を収束させる方法を取り入れたが、これは完全に制約を充足しなくても解を求める、という意味では問題を解決しているが、本来は重みを正しく設定することが本質的な問題の解決方法であり、今回はそれが達成できなかった。

テーマ選択に参考した文献

[1] 磯村 厚誌, 伊藤 孝行, 大囿 忠親, 新谷 虎松
ナーススケジューリングシステムにおける動的重み付き CSP に基づく再スケジューリング機能の実装
情報処理学会第 66 回全国大会