Тестовое задание

Цифровой планировщик

1. Введение

Одна из актуальных задач на производстве связана с построением оптимального графика выполнения технологических операций (обработка деталей, сборка узлов, отгрузка готовой продукции и пр.). Операции могут выполняться как параллельно, так и последовательно, на них могут быть наложены требования по крайнему сроку завершения. Перечень требований может изменяться динамически, поэтому необходимо разработать универсальный инструмент для планирования производственных процессов.

2. Описание задачи

2.1. Часть 1. Алгоритм построение графика

2.1.1. Имеется:

а. N задач и 1 станок

б. Каждая задача имеет крайний срок выполнения d_i, время выполнения t_i и награду p_i

в. Награда выдается за каждую задачу, выполненную в срок (т.е. $tstart_i + t_i <= d_i$)

г. Задачи могут выполняться в любом порядке, строго последовательно

д. Станок может выполнять только 1 задачу в каждый момент времени

2.1.2. Необходимо:

а. Разработать алгоритм поиска оптимального расписания выполнения задач (в виде последовательности номеров задач) с точки зрения максимизации награды

б. Формат входных и выходных данных гибкий (напр. csv)

в. Допускается использование внешних солверов (pulp, cvxopt)

г. Не допускается использование готовых решений

2.2. Часть 2. Универсальный планировщик

2.2.1. Имеется:

а. Динамический набор ограничений, т.е. перечень применяемых ограничений определяется в качестве входного параметра планировщика и неизвестен заранее

б. Перечень возможных динамических ограничений:

- Задача ***i*** может выполняться строго после задачи ***j***
- Задача **k** может нарушать крайний срок выполнения за незначительный штраф $\max(s_k) < \min(p_i)$
- Целевая функция $\text{sum}(p_i) - \text{sum}(s_i)$

2.2.2. Необходимо:

а. Разработать синтаксис универсального языка описания ограничений

б. Продумать архитектуру планировщика с учетом следующих возможностей:

. Возможность динамического определения ограничений задачи

. Возможность автоматического перестроения линейной модели исходя из ограничений

. Возможность подключения альтернативных солверов и алгоритмов оптимизации

. Возможность масштабирования планировщика на другие задачи (напр. планирование авиарейсов)

в. Разрабатывать сам универсальный планировщик не требуется

3. Ожидаемый результат

- Репозитарий (напр. на github) с реализацией алгоритма построения графика (часть 1), шаблон кода универсального планировщика (часть 2), отражающий его архитектуру, а также синтаксис языка описания динамических ограничений (с примерами),
- Репозитарий должен включать README.md с примером установки и запуска алгоритма построения графика, а также интерпретации результатов его работы на произвольных входных данных
- Репозитарий должен включать набор тестов для проверки корректности работы алгоритма построения графика (часть 1)
- Язык программирования любой (предпочтительно Python или C++)