CONSTRUCTIVE ALGORITHMS

Depth-first algorithm:

- Maak een rooster-stack
- Push een incompleet rooster op de stack (misschien beginnen met invullen totdat er te veel maluspunten te ontstaan?)
- Pop bovenste item van de stack
- Maak alle kinderen van dat item
- Compleet rooster? Check scores. Bewaar beste score

Nadelen:

- Hiermee worden vooral roosterconflicten en zaalconflicten opgelost en niet de andere problemen

SEARCH ALGORITHMS

Genetic algorithm:

- 15 chromosomen random opstellen als volgt: [rooster, score]
- 4 parent chromosomes selecteren door vier keer random 2 chromosomen te kiezen en daarvan de beste te selecteren
- Van de vier parent chromosomes kies je random de twee beste voor de cross-over
- Cross-over : dagen random wisselen
- Mutatie treedt met 10% kans bij child chromosoom op. Activiteiten worden in dat geval random gewisseld
- Child chromosome wordt gecheckt op feasibility:
    - Wordt weggegooid als hij niet feasible is
    - Child chromosome wordt een mogelijke oplossing als hij wel feasible is. Score van het child chromosome wordt berekend
- Herhaal bovenstaand proces tot er 8 child chromosomen zijn
- Vervang de 8 slechtste chromosomen in de parent populatie met de child chromosomes
- Herhaal bovenstaand proces voor bijvoorbeeld 50 generaties

Nadelen:

- Runtime. Not suitable for feasible solutions to realistically sized problems
- Work best when all constraints are modelled as “soft”

Tabu search:

- Variation of the hill climbing algorithm; attempts to solve problem of getting stuck in local optimum by allowing solutions to get worse under certain conditions and add them to the Tabu list

Nadelen:

- Extreme high computation times, because high number of banned positions needed to force a solution out of a local optimum

Hill climbing algorithm:

- Based on improving a solution by performing a single change (e.g. moving a single lesson) and accepting the change if the schedule remains valid and the quality improves
- Advantage of being relatively fast
- Algorithm is almost always used in combination with other search methods

Nadelen:

- Zeer waarschijnlijk dat je alleen lokaal optimum vindt. Optima found can almost always be improved by further calculations or by restarting the algorithm with new initial configuration.

Simulated annealing:

- Probabilistic process
- Comparable to Hill climbing algorithm: looking at neighbor solution
- As the algorithm progresses, the amount a schedule can get worse decreases
  - We keep a temperature variable to simulate the heating process
  - We initially set it high and then allow it to slowly cool as the algorithm runs
  - As the temperature is reduced, so is the chance of accepting worse solutions
- Ideal for estimating a global optimum in a large search space

3. Optimal & Heuristic Algorithms

Algorithms exist that can find an **optimal** solution (one which produces the minimum/maximum value of the objective function) in a "reasonable" time (relative to the size of the problem) for the very simplest schedule-related problems -- in particular, for simple versions of workforce scheduling and basic project scheduling.

However, for almost all of the schedule-related problems we've mentioned, optimal algorithms simply take too long, for all but but the smallest size problems. So for most of these problems, we need to use **heuristic algorithms**, which can produce solutions which are **near-optimal**, that is, reasonably close to the optimal solution, in a reasonable amount of time.

There are a number of common heuristic approaches to algorithms used for addressing schedule-related problems, which can be divided into

- **Construction Algorithms** (including **Greedy Algorithms**), which start with an empty or incomplete solution (e.g. where no tasks are scheduled and/or no resources are assigned), and incrementally make it more complete (e.g. by scheduling one additional task and/or assigning one additional resource at a time)

- **Search Algorithms,** which start with one or more complete candidate solutions, and incrementally combine and/or modify them with the goal of generating better complete solutions. These include approaches (also called metaheuristics) such as **Hill Climbing**, **Tabu Search**, **Simulated Annealing**, **Particle Swarm Optimization**, **Artificial Bee Colony Algorithms**, **Genetic Algorithms**, and **Differential Evolution**,.

- **Hybrid Algorithms,** which use a combination of construction and search-based approaches

Ellis Cohen
Jarurat Ousingsawat
Project.net