Justification of research and implementation

Daan Derks en Ferre van Oort

21-10-2024

Contents

[Problem definition 3](#_Toc179970121)

[Research approach 4](#_Toc179970122)

[Design heuristic based on Improving Search 5](#_Toc179970123)

[Python implementation 6](#_Toc179970124)

[Tests and experiments 7](#_Toc179970125)

[Visualizations 8](#_Toc179970126)

[Realized depth in OR 9](#_Toc179970127)

# Problem definition

* Scheduling problem
* Objective:  
  Minimize both setup times (from color switches) and penalties (from late orders) when scheduling car part painting on multiple machines.
* Input:
  + Orders (O): Each order o***i​*** has:
    - Surface Area(A***i​***​): Determines processing time based on machine speed.
    - Color (C***i​***​): Orders need specific colors; switching colors incurs setup time.
    - Deadline (D***i​***): Time by which the order must be completed.
    - Penalty (P***i​***​): Cost per unit of lateness if t***end*** > D***i***
  + Machines (M):  
    Each machine M***j​*** has:
    - Speed (S***j​***​): Determines the time required to paint an order.
    - Setup Time (T***xy***​): Time to switch from color C***x​***​ to C***y***.
* Solution:  
  A schedule S that assigns orders to machines, specifying:
* Start Time t***start*** ​, End Time t***end***​, and Color for each order.
* Minimize total penalties and setup times for each machine.
* Constraints:
* Each machine processes one order at a time, no partial processing allowed.
* Orders must respect machine availability and deadlines.
* Objective Function: Minimize:

Where is a weighting factor to balance penalties and setup costs.

* Feasible Solution:  
  A valid assignment of orders to machines with respect to deadlines, processing times, and color setup constraints.

A close-up of a piece of paper

Description automatically generatedA close-up of a piece of paper

Description automatically generated

# Research approach

* Literature
  + R. Rardin (2014), *Optimization in Operations Research*, Pearson New International Edition
  + E.H.L. Aarts and J.K. Lenstra (Editors) (2003), *Local Search in Combinatorial Optimization*, Princeton University Press
* Use of external sources
  + ChatGPT-4o was used as help by implementing the solution method in Python
    - Brainstorming on schedule data-structures
    - Obtaining Python code snippets for improving search heuristics
    - Advice on the best metaheuristic to use

Source: ()

# Design heuristic based on Improving Search

* Chosen neighborhood structure
  + Move: Swap 2 orders on the same or a different machine to search for potential improvements
* Heuristic solution methods
  + Constructive heuristic
    - Start with an initial schedule by assigning orders based on deadlines and availability
    - Ensure minimal colour switches and setup times
  + 2-opt Improving search
    - Iteratively swap 2 orders and evaluate the new schedule
    - If the swap lowers total penalty and setup time, accept the new schedule
    - Repeat until no further improvement
  + Tabu search
    - Iteratively swap 2 orders and evaluate the new schedule
    - Even if they temporarily worsen the solution, accept the new schedule
    - Use a Tabu List to avoid revisiting recent solutions
    - Continue search until a (near) optimal solution is reached
* Discrete improving search logic
* Tabu search logic

# Python implementation

We began testing our code with 3 machines and 3 orders to keep it simple and added complexity along the way.

* Orders, set of all orders O = {1, 2, 3}
* Machines, set of all machines M = {1, 2, 3}
* Functions defined:
  + Painttime(area, machine, machines)
    - Returns the time for machine M to paint an area
  + Switchtime(prevcolor, currentcolor)
    - Returns the time for any machine M to switch colors from prevcolor to currentcolor
  + Schedule\_orders(orders, machines)
    - Returns a schedule S for all orders O on machines M
  + Calculate\_penalty()
    - Returns the sum of the penalties for all orders O on schedule S
  + Swap\_orders\_optimization(orders, machines, max\_iterations)
    - Returns schedule S after optimization
* Organisation of the code
  + There is one file containing both the implementation of the Discrete Improving Search and the Tabu Search
  + Discrete Improving Search is implemented in function swap\_orders\_optimizations
  + Tabu Search is implemented in function

# Tests and experiments

* Started with a simplified version with only one machine and no deadlines and penalties
* Started with a ChatGPT snippet for scheduling the orders
  + Double-checked implementation of switchtime: NOK
    - Didn’t use the right syntax for variable type
  + Double-checked implementation of painttime: NOK
    - Didn’t use the right syntax for variable type
* Implemented function to calculate the penalties of all orders
* Added orders to test function calculate\_penalty()

# Visualizations

# Realized depth in OR

* Implemented Constructive heuristic
* Implemented 2-opt Improving Search
* Implemented Tabu Search
* Compared performance between all implementations based on lowest penalty