

1 Scheduling Model

1.1 Necessary Data

1.1.1 Description

The data is split into 2 parts:

- System-Info, contains information about the production environment
- Orders, contains orders made by customers, which need to be processed in the described production environment

1.1.1.1 System-Info

Contains information about the production environment

1.1.1.1.1 Tasks All steps / actions, that can be performed in the production environment. (Important: Tasks which are added as "follow up task" to a task should NOT have this task added as "preciding task"). The duration of each task can be found in the data of the workstations (paragraph 1.1.1.1.4), so the processing of different tasks on different machines/workstations can be accounted for.

- id – unique ID of the task (just used for internal identification)
- name – name of the task (for easier readable/interpretable results)
- resources – (list) needed resources, to process the task (id + amount)
- products – (list) all resources (id + amount) which are created by completing the task
- preceding_tasks – (list) tasks, which need to be scheduled before this task can be done
- follow_up_tasks – (list) tasks, which need to be scheduled after this task is completed
- independent – boolean, to flag if the task should be used in the schedule on it's own (e.g. false for tasks which only ever appear as "preciding tasks" and wouldn't make sense on their own)
- prepare_time – the time needed to prepare the machine/workstation for this task
- unprepare_time – the time needed to unprepare a machine/workstation after it was used for this task

1.1.1.1.2 Recipes The recipes for the different resources, which are supposed to be produced (final products or intermediate products)

- id – unique ID of the recipe (just used for internal identification)
- name – name of the recipe (for easier readable/interpretable results)
- tasks – (list) IDs of all tasks, which need to be scheduled/processed, to finish this recipe (result resources and amounts can be looked up from the result products of the last task in the recipe)

1.1.1.1.3 Resources All resources, which can either be used or produced in the production environment

- id – unique ID of the resource (just used for internal identification)
- name – name of the resource (for easier readable/interpretable results)
- stock – amount of this resource currently in stock
- price – price per unit to buy this resource (0, if the resource can not be bought)
- delivery_time - expected delivery time, if the resource has to be bought
- renewable – boolean, to flag if the resource is consumed or not (e.g. a resource representing an employee is renewable, because they are usable again once a task is done)
- recipes – (list) IDs of the recipes, with which this resource can be produced (if there are any, otherwise empty list)

1.1.1.1.4 Workstations All machines/workstations, which are available in the production environment

- id – unique ID of the workstation (just used for internal identification)
- name – name of the workstation (for easier readable/interpretable results)
- basic_resources – (list) contains the IDs of the necessary resources + amount, which will be needed to run the workstation (independent of the resources needed to process a task)
- tasks – (list) IDs of all tasks, which can be processed on this workstation + the duration of each task on this workstation

1.1.1.2 Orders

Describes orders made by customers, which are supposed to be processed in the production environment.

- `id` – unique ID of the order (just used for internal identification)
- `arrival_time` – timestamp, when the order was made
- `delivery_time` – timestamp, when the order is supposed to be done
- `latest_acceptable_time` – timestamp, latest acceptable point in time to which the order needs to be completed if the given delivery time can not be met
- `resources` – (list) IDs + amounts of ordered resources, + the price the customer pays for each of the resources
- `penalty` – penalty if the order is not processed
- `tardiness_fee` – fee, if the order is only completed after the delivery time but before the latest acceptable time
- `divisible` – boolean, to flag if the order can be split and only partially fulfilled or if all of the ordered resources have to be done (or none)
- `customer_id` – unique ID of the customer
- `optional` - boolean, flags if the order has to be scheduled or if it is optional (default: true)

1.1.2 Jobs

Describes a specific instance of a task for an order.

- `id` - unique ID of the job (just used for internal identification)
- `order_id` - ID of the order, for which the job is being scheduled
- `task_id` - ID of the task which is processed
- `to_id` - specific assignment ID between the tasks and the orders to make it easier to link specific tasks to orders

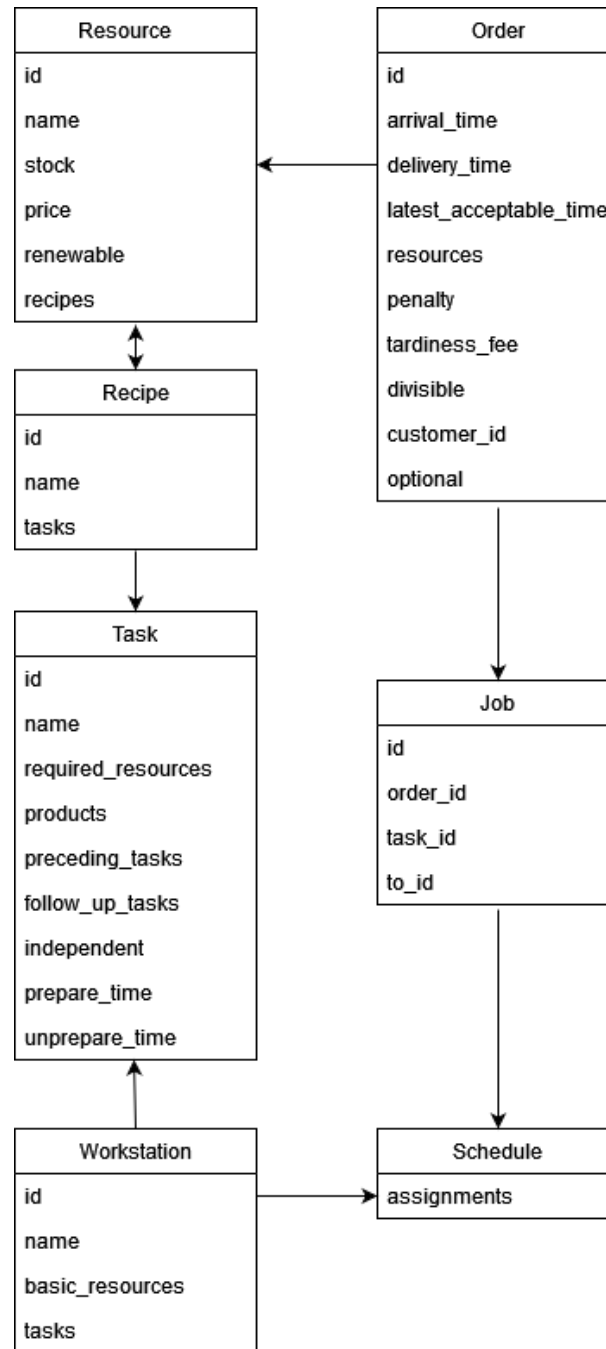


Figure 1.1: Dependencies

1.2 Input / Output

Exact structure of the data for with example data can be seen at "JSON-Example" (section 1.5).

Input:

Orders:

OrderA

$Task_1 - Task_n$

$price_1 - price_n$

penalty

tardiness_fee

delivery_date

latest_date

optional

Mapping of tasks to jobs example::

TaskA.1

TaskID

JobID

Intermediate:

<Workstation ID, Task (Operation) ID, Start Time Slot> $j_1 - (w, t, s)$

$j_2 - (w, t, s)$

$j_2 - (w, t, s)$

.

.

.

$j_n - (w, t, s)$

Output:

$$\begin{bmatrix} (j_{11}, s_{11}) & (j_{12}, s_{12}) & (j_{13}, s_{13}) & \dots & (j_{1n}, s_{1n}) \\ (j_{21}, s_{21}) & (j_{22}, s_{22}) & (j_{23}, s_{23}) & \dots & (j_{2n}, s_{2n}) \\ \dots & \dots & \dots & \dots & \dots \\ (j_{m1}, s_{m1}) & (j_{m2}, s_{m2}) & (j_{m3}, s_{m3}) & \dots & (j_{mn}, s_{mn}) \end{bmatrix}$$

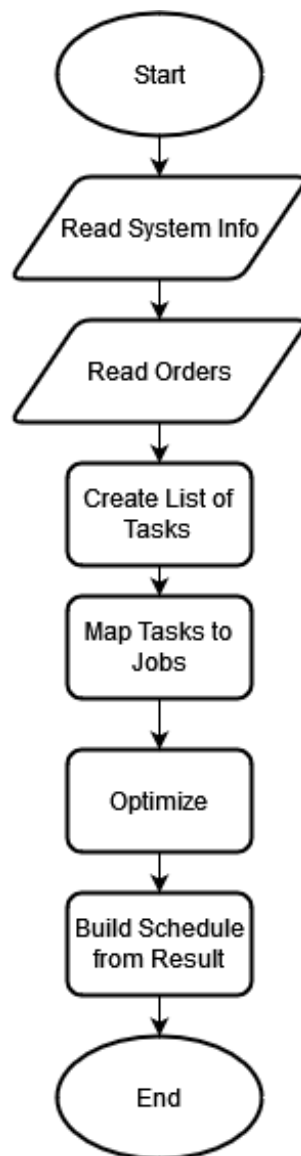


Figure 1.2: Ablauf

1.3 Model

1.3.1 Variable Definition

OLD:

O - Set of all Orders

o - Specific Order, $o \in O$

W - Set of all Workstations

T - Set of all Tasks
 T_o - Set of all Tasks needed for Order o , $T_o \subset T$
 x - Specific Task, $x \in T$
 T_{xp} - Set of all Tasks preceding Task x , $T_{xp} \subset T$
 T_{xf} - Set of all Tasks following Task x , $T_{xf} \subset T$
 J - Set of all Jobs
 J_w - Set of all Jobs assigned to Workstation w , $w \in W$
 j_x - Job linked to Task j , $x \in T$, $j \in J$
 x_j - Task linked to Job j , $x \in T$, $j \in J$
 W_j - Set of Workstations eligible for Job j , $W_j \subset W$
 R - Set of all Resources
 w - Specific Workstation $w \in W$
 R_j - Set of Resources needed for Job j , $R_j \subset R$
 r - Specific Resource, $r \in R$
 d_{xw} - Duration of Task x on Workstation w
 j_w - Selected Workstation for a Job j , $w \in W_j$
 j_s - Start time slot of Job j
 j_{dt} - Deliver time slot of Job j
 j_{lt} - Latest allowed time slot of Job j
 i_{rt} - Amount of Resource r in the Inventory at time slot t
 l_j - Binary Variable, 1 if Job j is late (Tardiness Fee applies)
 l_o - Binary Variable, 1 if Order o is late (Tardiness Fee applies)
 u_o - Binary Variable, 1 if Order o can only be fulfilled partially
 p_o - Price the customer pays for Order o
 f_o - Price reduction for tardy orders
 O_c - Set of all Orders which could be completed in the derived schedule,
 $O_c \subset O$

NEW SUGGESTION:

O - Set of all Orders
 P - Set of all Tasks
 W - Set of all Workstations
 R - Set of all Resources
 X - Set of all Jobs
 o - Specific Order, $o \in O$
 p - Specific Task, $p \in P$
 w - Specific Workstation $w \in W$
 r - Specific Resource, $r \in R$
 x - Specific Job, $x \in X$
 P_o - Set of all Tasks needed to complete Order o , $P_o \subset P$

P_{pre} - Set of all Tasks preceding Task p , $P_{pre} \subset P$
 P_{pf} - Set of all Tasks following Task p , $P_{pf} \subset P$
 X_w - Set of all Jobs assigned to Workstation w , $w \in W$, $X_w \subset X$
 W_x - Set of all Workstations eligible for Job x , $x \in X$, $W_x \subset W$
 R_x - Set of all Resources required for Job x , $x \in X$, $R_x \subset R$
 d_{pw} - Duration of Task p on Workstation w , $p \in P$, $w \in W$, $d_{pw} \geq 0$
 x_w - Selected Workstation w for Job x , $x \in X$, $w \in W$, $x_w \in W$
 x_s - Start time slot for Job x , $x \in X$, $x_s \geq 0$
 x_p - Task p corresponding to Job x , $x \in X$, $p \in P$, $x_p \in P$
 x_{dt} - Delivery time slot for Job x , $x \in X$, $x_{dt} \geq 0$
 x_{lt} - Latest acceptable time slot for Job x , $x \in X$, $x_{lt} \geq 0$, $x_{lt} \geq x_{dt}$
 $a_{r,t}$ - Amount of Resource r in stock at time slot t , $r \in R$, $t \geq 0$, $a_{r,t} \geq 0$
 l_x - Binary Variable, 1 if Job x is late (Tardiness Fee applies), $x \in X$
 l_o - Binary Variable, 1 if Order o is late (Tardiness Fee applies), $o \in O$
 u_o - Binary Variable, 1 if Order o can only be fulfilled partially, $o \in O$,
 z_o - Price the customer pay for Order o , $o \in O$, $z_o \geq 0$
 f_o - Price reduction for tardy orders, $o \in O$, $f_o \geq 0$
 o_{opt} - Binary Variable, 1 if Order o is optional, $o \in O$
 o_{lt} - Latest acceptable time for Order o , $o \in O$, $o_{lt} \geq 0$
 b_o - Penalty, if Order o is not scheduled to meet o_{lt} , $o \in O$, $b \geq 0$
 O_c - Set of all Orders which could be completed in the schedule, $O_c \subset O$

1.3.2 Objective Function + Constraints

1.3.2.1 Objective Functions

Equation 1 Minimize tardy jobs

Equation 2 Maximize earning

Equation 3 Minimize deviation from the expected delivery date

Equation 4 Maximize Set Size of scheduled jobs

Equation 5 Maximize Set Size of scheduled jobs with least tardy jobs

Equation 6 Maximize Set Size of scheduled jobs with least deviations

$$\begin{aligned}
 X &:= X_i \\
 \text{minimize } &\sum_{i=0}^X l_x
 \end{aligned} \tag{1}$$

$$\begin{aligned}
o &:= O_{c,i} \\
\text{maximize } & \sum_{i=0}^{O_c} p_o - (l_o * f_o)
\end{aligned} \tag{2}$$

$$\begin{aligned}
x &:= X_i \\
y &:= p_j \\
\text{minimize } & \sum_{i=0}^X |(x_s + d_{y,w}) - x_{dt}|
\end{aligned} \tag{3}$$

$$\text{maximize } |O_c| \tag{4}$$

$$\begin{aligned}
x &:= X_i \\
\text{maximize } & |O_c| - \sum_{i=0}^X l_x
\end{aligned} \tag{5}$$

$$\begin{aligned}
x &:= X_i \\
y &:= p_x \\
z & \dots \text{threshold for acceptable deviation} \\
d &:= \sum_{i=0}^X (|(x_s + d_{y,w}) - x_{dt}| > z) \\
\text{maximize } & |O_c| - d
\end{aligned} \tag{6}$$

1.3.2.2 Constraints

Equation 7 to make sure jobs finish before last possible time slot

Equation 8 checks if tardiness fee applies

Equation 9 starting times need to be after timeslot 0

Equation 10 to make sure jobs have sufficient resources to start

Equation 11 jobs can't start before preceding tasks finish

Equation 12 follow up jobs can't start before job finished

Equation 13 only one active job at one time slot for each Workstation

UPDATED:

$$x_s \leq x_{lt} - d_{x_p, x_w} \quad (7)$$

$$l_x := x_{dt} \leq x_s \leq x_{lt} \quad (8)$$

$$\begin{aligned} x &:= X_i \\ \sum_{i=0}^X (x_s < 0) &< 1 \end{aligned} \quad (9)$$

$$\begin{aligned} r &:= R_{xi} \\ \sum_{i=0}^{R_x} (a_{r,t} - r \leq 0) &< 1 \end{aligned} \quad (10)$$

$$\begin{aligned} y &:= P_{x_{pre}, i} \\ y' &:= P_{x_{pre}, i-1} \\ z &:= \sum_{i=1}^{P_{x_{pre}}} (x_{ys} + d_{yw}) - (x_{y'} + d_{y'w}) < 0 \\ z &< 1 \end{aligned} \quad (11)$$

$$\begin{aligned} y &:= P_{x_{pf}, i} \\ y' &:= P_{x_{pf}, i-1} \\ z &:= \sum_{i=1}^{P_{x_{pf}}} (x_{y's} + d_{y'w}) < x_{ys} \\ z &< 1 \end{aligned} \quad (12)$$

NOT YET UPDATED:

$$\begin{aligned}
&\forall w \in W \\
&i = J_{wm} \\
&j = J_{wn} \\
&x = x_i \\
&x' = x_j
\end{aligned} \tag{13}$$

$$\sum_{m=0}^{J_w} \left(\sum_{n=0}^{J_w-1} ((1 - (j_s + d_{x',w} < i_s)) + (1 - (j_s > i_s + d_{x,w}))) \right) < 1$$

1.4 Tests & Experiments

1.4.1 Currently used Objective Functions

1.4.1.1 Make-span

$$\begin{aligned}
&x_{end,p} = x_s + d_{pw} \\
&p_{first...firstscheduledtask} \\
&p_{last...lastscheduledtask} \\
&minimize x_{end,p_{last}} - x_{s,p_{first}}
\end{aligned} \tag{14}$$

1.4.1.2 Min Deadline Deviation

$$\begin{aligned}
&x := X_i \\
&y := p_j \\
&minimize \sum_{i=0}^X |(x_s + d_{y,w}) - x_{dt}|
\end{aligned} \tag{15}$$

1.4.1.3 Min Idle Time

$$\begin{aligned}
&x := w_i \\
&x' := w_{i-1} \\
&w := W_j \\
&\sum_{j=0}^W w_{0s} + \sum_{i=1}^w x_s - (x'_s + d_{x'_p w})
\end{aligned} \tag{16}$$

1.4.2 Genetic Algorithm Approach

Tested own implementation and PyGAD library - currently using PyGAD library (same results, but faster runtime)

1.4.2.1 Feasibility Function

The current version of the feasibility function tests the following cases:

- start times $<$ earliest allowed time slot
- end times $>$ last allowed time slot
- overlap of jobs scheduled on the same workstation
- correct sequence of jobs for each order (including no overlap)

1.4.2.2 Encoding

There are currently two different encodings in use, depending on which strategy is being used. The 1-Stage Optimization attempts to optimize both the workstation assignment and the start times of each job at the same time, while the 2-Stage Optimization handles the assignments to the workstations first and only after this finished, the start times are adapted to create a feasible schedule.

1.4.2.2.1 1-Stage Optimization For every Job which needs to be scheduled, on pair of $\langle \text{workstation}, \text{start time} \rangle$ is used - the duration for each job can be looked up using the workstation and the type of Job for each gene.

1.4.2.2.2 2-Stage Optimization In Stage 1, every Job is represented by a single integer $\langle \text{workstation} \rangle$ In Stage 2, every Job is represented by a single integer $\langle \text{start time} \rangle$ - the workstation assignments needed for the fitness function are taken of the result of Stage 1

1.4.2.3 Selection

The selection method used for all current experiments / tests was the "Roulette Wheel Selection", both with the PyGAD library and the self implemented GA version.

1.4.2.4 Recombination

On the used scale of complexity in the chosen test problems for scheduling, there was no significant difference in the performance of the two tested recombination methods. Once the GA manages to find any feasible solution, both methods keep improving the solution.

1.4.2.4.1 One-Point-Crossover For the used One-Point-Crossover Variant, 2 parents are selected, a random point between 0 and n is chosen ($n = \text{amount of genes} - 1$) and 2 new offsprings are created.

1.4.2.4.2 Two-Point-Crossover For the used Two-Point-Crossover Variant, 2 parents are selected, two random points between 0 and n is chosen ($n = \text{amount of genes} - 2$) and between Point 1 and m ($m = \text{amount of genes}$) and 2 new offsprings are created.

1.4.2.5 Mutation

1.4.2.5.1 Random The random mutation randomly chooses genes of an individual and changes them to any of the available options for the respective gene (e.g. any time slot between the earliest allowed and last allowed time slot)

1.4.2.5.2 Only Feasible Mutations This mutation method also randomizes the genes of an individual, but only picks from the pool of feasible combinations (task-workstation assignment).

1.4.2.6 Manual Result Improvement

At the end of the optimization using a GA, the order and assignments of the Jobs may already be good, but some unnecessary unused space is either in front of all jobs on a workstation or in between jobs on a workstation. To circumvent this, an algorithm to improve the result is used, which removes the unused space and shifts all jobs closer together (without violating constraints). The algorithm currently used works like this:

Algorithm 1 Result Compression Algorithm

```
new_solution = deepcopy(solution)
changes_made = True
while changes_made do
    changes_made = False
    for all workstations w in workstations do
        sort jobs on workstation w by start time
        for Job j in sorted jobs of workstation w do
            if j is the first job in order sequence then
                shift = 0
                if j is the first job on current workstation then
                    shift = new_solution[j] - earliest_allowed_time
                else
                    shift = new_solution[j] - new_solution[j - 1] + durations[j - 1]
                end if
            else
                previous_job_end = new_solution[j - 1] + durations[j - 1]
                max_shift = new_solution[j] - previous_job_end
                if j is the first job on current workstation then
                    shift = max_shift
                else
                    shift = new_solution[j] - new_solution[j - 1]
                    if max_shift < shift then
                        shift = max_shift
                    end if
                end if
            end if
            new_solution[j] = new_solution[j] - shift
            changes_made = shift > 0 OR changes_made
        end for
    end for
end while
```

This algorithm currently does not scale well for higher complexities, but has a lot of potential for improvement. The algorithm iteratively improves the schedule until there are no more (legal) changes to be made.

1.4.3 Agent-Based Approach

The current Agent-Based Approach is built like an Autonomous System and works with a simplistic strategy (greedy) just to demonstrate how such a solution could work. It currently simulates the process of continually feeding new orders into the system by processing them one at a time. To test a hybrid approach, a second version of the same agent exists, for which a GA tries to optimize the sorting of the incoming orders to improve the result. A second version (currently untested) is also implemented, which in addition to the greedy assignment strategy, follows a secondary strategy for cases in which a job can not be scheduled in a good (or just valid) timeslot, to make sure more complex problems could be solved by the approach as well.

TODO: There are different approaches to the agent-based approach, where either all agents perform the full optimization and the results are compared, and where the optimization with each agent is performed step by step and the population for all is chosen from all agents. (skeletons for both are implemented)

Additionally, every other approach can be implemented as agent-based solution and be compatible to work with a multi-agent system. Reinforcement learning and other machine learning methods can also be applied to the agent-based approach.

1.4.3.1 3-Stage Agent-Based Optimization

Based on: <http://link.springer.com/10.1007/s40092-017-0204-z>

The described scheduling system consists of 3 parts, which can be implemented using 3 agents working in sequence. First, a GA, using an adapted crossover method based on iPOX, creates a population of feasible schedules. For the mutation of the workstation assignments, a random assignment is chosen and replaced with another workstation which can be used for the given task. The mutation of the operation order is conducted by randomly selecting 2 indices and swapping the respective selected operations. In the second step, the found solutions in the result population of the GA are clustered to find the different groups of solutions. The third agent performs a local search in each cluster to find the best solution from the given suggestions (in the paper, the local search method used is Tabu Search, other examples use e.g. Harmony Search). The best solution is chosen from the search result of each cluster.

1.4.4 CMA-ES

The used CMA-ES library can not differ between multiple gene types, so currently, the only comparison can be done with the 2-Stage Optimization. In the first stage, the workstation assignments are done while in the second stage the start times are optimized.

1.4.5 Comparison

To compare the different approaches, multiple evaluation functions have been implemented. Since every optimization method can produce a schedule according to the data model, the functions can evaluate the results of all methods. The currently implemented comparison functions are:

- Makespan - Time needed to complete all scheduled tasks from first scheduled task to the end of the last scheduled task
- Tardiness - Sum of all delays between delivery times and last acceptable times
- Deviation - Sum of all schedule deviations from delivery times
- Idle-Time - Sum of all idle times on each workstation
- Profit - Sum of all profits over all scheduled orders - penalties and fees for delays and orders not being processed at all

1.4.5.1 Data-Sets

The data sets used to compare the different approaches are taken from various sources:

- <https://data.mendeley.com/datasets/hpp82wtxfr/1> which provides the data originally used by Gong et al., 2017
- <https://github.com/tamy0612/JSPLIB> which combines multiple data sets from different sources (found in the repository description).
- <https://github.com/MiniZinc/minizinc-benchmarks> which provides datasets from the MiniZinc challenges (2008-2012)
- There are four more instances taken from Kacem et al. (2002b), and some smaller made up testing instances.

1.4.6 Currently missing

:

- Optimization with regard to resource constraints
- Predictions included into the optimization process (missing data), e.g. workstation maintenance, resource delivery, expected workloads/orders, ...
- Well structured re-implementation of useful comparison methods

1.5 JSON-Example

```
1 {
2   "system-info":
3     {
4       "tasks":
5         [
6           {
7             "id": 0,
8             "name": "Task 1",
9             "resources":
10              [
11                {
12                  "id": 0,
13                  "amount": 10
14                }
15              ],
16             "products":
17              [
18                {
19                  "resource_id":
20                    1,
21                  "amount": 1
22                }
23              ],
24             "preceding_tasks":
25              [],
26             "follow_up_tasks":
27              [
```

```

27         1
28     ],
29     "independent": true,
30     "prepare_time": 5,
31     "unprepare_time": 5
32 },
33 {
34     "id": 1,
35     "name": "Task 2",
36     "resources":
37     [
38         {
39             "id": 1,
40             "amount": 1
41         }
42     ],
43     "products":
44     [
45         {
46             "resource_id":
47                 2,
48             "amount": 1
49         }
50     ],
51     "preceding_tasks":
52     [],
53     "follow_up_tasks":
54     [],
55     "independent": false,
56     "prepare_time": 0,
57     "unprepare_time": 5
58 },
59 {
60     "id": 2,
61     "name": "Task 3",
62     "resources":
63     [
64         {
65             "id": 3,
66             "amount": 20

```

```

66         },
67         {
68             "id": 4,
69             "amount": 10
70         }
71     ],
72     "products":
73     [
74         {
75             "resource_id":
76                 5,
77             "amount": 2
78         },
79         {
80             "resource_id":
81                 6,
82             "amount": 1
83         }
84     ],
85     "preceding_tasks":
86     [
87         3, 4
88     ],
89     "follow_up_tasks":
90     [],
91     "independent": true,
92     "prepare_time": 10,
93     "unprepare_time": 5
94 },
95 {
96     "id": 3,
97     "name": "Task 4",
98     "resources":
99     [
100         {
101             "id": 7,
102             "amount": 5
103         }
104     ],
105     "products":

```

```

104         [
105             {
106                 "resource_id":
107                     3,
108                 "amount": 20
109             }
110         ],
111         "preceding_tasks":
112         [],
113         "follow_up_tasks":
114         [],
115         "independent": false,
116         "prepare_time": 5,
117         "unprepare_time": 5
118     },
119     {
120         "id": 4,
121         "name": "Task 5",
122         "resources":
123         [
124             {
125                 "id": 8,
126                 "amount": 1
127             }
128         ],
129         "products":
130         [
131             {
132                 "resource_id":
133                     4,
134                 "amount": 5
135             }
136         ],
137         "preceding_tasks":
138         [],
139         "follow_up_tasks":
140         [],
141         "independent": true,
142         "prepare_time": 1,
143         "unprepare_time": 5

```

```

142     },
143     {
144         "id": 5,
145         "name": "Task 6",
146         "resources":
147         [
148             {
149                 "id": 9,
150                 "amount": 20
151             },
152             {
153                 "id": 10,
154                 "amount": 1
155             }
156         ],
157         "products":
158         [
159             {
160                 "resource_id":
161                     8,
162                 "amount": 2
163             }
164         ],
165         "independent": true,
166         "preceding_tasks":
167         [],
168         "follow_up_tasks":
169         [],
170         "prepare_time": 5,
171         "unprepare_time": 5
172     }
173 ],
174 "recipes":
175 [
176     {
177         "id": 0,
178         "name": "Recipe 1",
179         "tasks":
180         [
181             5

```

```

181     ]
182     },
183     {
184         "id": 1,
185         "name": "Recipe 2",
186         "tasks":
187         [
188             2
189         ]
190     },
191     {
192         "id": 2,
193         "name": "Recipe 3",
194         "tasks":
195         [
196             0
197         ]
198     },
199     {
200         "id": 3,
201         "name": "Recipe 4",
202         "tasks":
203         [
204             4
205         ]
206     }
207 ],
208 "resources":
209 [
210     {
211         "id": 0,
212         "name": "Resource 1",
213         "stock": 50,
214         "price": 20,
215         "delivery_delay": 0,
216         "renewable": false,
217         "recipes":
218         []
219     },
220     {

```



```

221         "id": 1,
222         "name": "Resource 2",
223         "stock": 0,
224         "price": 300,
225         "delivery_delay": 0,
226         "renewable": false,
227         "recipes":
228         []
229     },
230     {
231         "id": 2,
232         "name": "Resource 3",
233         "stock": 10,
234         "price": 1000,
235         "delivery_delay": 0,
236         "renewable": false,
237         "recipes":
238         [
239             2
240         ]
241     },
242     {
243         "id": 3,
244         "name": "Resource 4",
245         "stock": 100,
246         "price": 10,
247         "delivery_delay": 0,
248         "renewable": false,
249         "recipes":
250         []
251     },
252     {
253         "id": 4,
254         "name": "Resource 5",
255         "stock": 10,
256         "price": 100,
257         "delivery_delay": 0,
258         "renewable": false,
259         "recipes":
260         [

```

```

261                                     3
262                                 ]
263         },
264         {
265             "id": 5,
266             "name": "Resource 6",
267             "stock": 4,
268             "price": 50,
269             "delivery_delay": 0,
270             "renewable": false,
271             "recipes":
272             [
273                 1
274             ]
275         },
276         {
277             "id": 6,
278             "name": "Resource 7",
279             "stock": 3,
280             "price": 70,
281             "delivery_delay": 0,
282             "renewable": false,
283             "recipes":
284             [
285                 1
286             ]
287         },
288         {
289             "id": 7,
290             "name": "Resource 8",
291             "stock": 15,
292             "price": 15,
293             "delivery_delay": 0,
294             "renewable": false,
295             "recipes":
296             []
297         },
298         {
299             "id": 8,
300             "name": "Resource 9",

```

```

301         "stock": 6,
302         "price": 700,
303         "delivery_delay": 0,
304         "renewable": false,
305         "recipes":
306         [
307             0
308         ]
309     },
310     {
311         "id": 9,
312         "name": "Resource 10",
313         "stock": 40,
314         "price": 200,
315         "delivery_delay": 0,
316         "renewable": false,
317         "recipes":
318         []
319     },
320     {
321         "id": 10,
322         "name": "Resource 11 (Employee
323             )",
324         "stock": 10,
325         "price": 0,
326         "delivery_delay": 0,
327         "renewable": true,
328         "recipes":
329         []
330     },
331     {
332         "id": 11,
333         "name": "Resource 12",
334         "stock": 1000,
335         "price": 5,
336         "delivery_delay": 0,
337         "renewable": false,
338         "recipes":
339         []
340     }

```

```

340     ],
341     "workstations":
342     [
343         {
344             "id": 0,
345             "name": "Workstation 1",
346             "basic_resources":
347             [
348                 {
349                     "id": 10,
350                     "amount": 1
351                 }
352             ],
353             "tasks":
354             [
355                 {
356                     "task_id": 0,
357                     "duration": 10
358                 },
359                 {
360                     "task_id": 2,
361                     "duration": 5
362                 }
363             ]
364         },
365         {
366             "id": 1,
367             "name": "Workstation 2",
368             "basic_resources":
369             [
370                 {
371                     "id": 10,
372                     "amount": 1
373                 },
374                 {
375                     "id": 11,
376                     "amount": 2
377                 }
378             ],
379             "tasks":

```

```

380     [
381         {
382             "task_id": 1,
383             "duration": 20
384         },
385         {
386             "task_id": 2,
387             "duration": 7
388         }
389     ]
390 },
391 {
392     "id": 2,
393     "name": "Workstation 3",
394     "basic_resources":
395     [],
396     "tasks":
397     [
398         {
399             "task_id": 3,
400             "duration": 1
401         },
402         {
403             "task_id": 4,
404             "duration": 2
405         },
406         {
407             "task_id": 5,
408             "duration": 15
409         }
410     ]
411 },
412 {
413     "id": 3,
414     "name": "Workstation 4",
415     "basic_resources":
416     [
417         {
418             "id": 10,
419             "amount": 1

```

```

420     }
421     ],
422     "tasks":
423     [
424         {
425             "task_id": 4,
426             "duration": 1
427         },
428         {
429             "task_id": 5,
430             "duration": 10
431         }
432     ]
433 }
434 ]
435 },
436 "orders":
437 [
438     {
439         "id": 0,
440         "arrival_time": "2022-02-07 00:00:00",
441         "delivery_time": "2022-02-20 14:30:00"
442         ,
443         "latest_acceptable_time": "2022-02-22
444             00:00:00",
445         "resources":
446         [
447             {
448                 "id": 2,
449                 "amount": 5,
450                 "price": 700
451             },
452             {
453                 "id": 4,
454                 "amount": 10,
455                 "price": 300
456             }
457         ],
458         "penalty": 300,
459         "tardiness_fee": 100,

```

```

458         "divisible": false,
459         "customer_id": 1,
460         "optional": true
461     },
462     {
463         "id": 1,
464         "arrival_time": "2022-02-17 00:00:00",
465         "delivery_time": "2022-02-21 15:30:00"
466         ,
467         "latest_acceptable_time": "2022-02-24
468             00:00:00",
469         "resources":
470         [
471             {
472                 "id": 4,
473                 "amount": 20,
474                 "price": 600
475             },
476             {
477                 "id": 5,
478                 "amount": 10,
479                 "price": 2000
480             }
481         ],
482         "penalty": 500,
483         "tardiness_fee": 100,
484         "divisible": false,
485         "customer_id": 1,
486         "optional": true
487     },
488     {
489         "id": 2,
490         "arrival_time": "2022-02-10 00:00:00",
491         "delivery_time": "2022-02-19 14:00:00"
492         ,
493         "latest_acceptable_time": "2022-02-26

```

```

494         "id": 2,
495         "amount": 20,
496         "price": 2700
497     },
498     {
499         "id": 5,
500         "amount": 10,
501         "price": 2000
502     }
503 ],
504 "penalty": 1000,
505 "tardiness_fee": 100,
506 "divisible": false,
507 "customer_id": 1,
508 "optional": true
509 },
510 {
511     "id": 3,
512     "arrival_time": "2022-02-08 00:00:00",
513     "delivery_time": "2022-02-23 17:00:00"
514     ,
515     "latest_acceptable_time": "2022-02-26
516         00:00:00",
517     "resources":
518     [
519         {
520             "id": 5,
521             "amount": 17,
522             "price": 3500
523         },
524         {
525             "id": 6,
526             "amount": 10,
527             "price": 1200
528         }
529     ],
530     "penalty": 700,
531     "tardiness_fee": 200,
532     "divisible": false,
533     "customer_id": 1,

```



```

532         "optional": true
533     },
534     {
535         "id": 4,
536         "arrival_time": "2022-02-15 00:00:00",
537         "delivery_time": "2022-02-23 12:00:00"
538         ,
539         "latest_acceptable_time": "2022-02-26
540             00:00:00",
541         "resources":
542         [
543             {
544                 "id": 4,
545                 "amount": 34,
546                 "price": 800
547             },
548             {
549                 "id": 8,
550                 "amount": 10,
551                 "price": 3000
552             }
553         ],
554         "penalty": 200,
555         "tardiness_fee": 100,
556         "divisible": false,
557         "customer_id": 1,
558         "optional": true
559     }
560 ]
561 }

```