**1 Context**

This stage is a simplified version of the final product, which will be used as a checkpoint. We need to show MouldMen that we are making progress.

**2 Summary**

Stage 1 will

* only assign one technician to each job
* have depots
* use technician availability
* use travel time

**3 Task**

Stage 1 will be a technician allocation algorithm for already scheduled jobs. We must find the most optimal solution for technicians to complete all the jobs.

A solution is a set of routes that the technicians can use to complete all the jobs.

The most optimal solution is the solution with the **smallest travel time**. If two solutions have the same travel time, select the solution with the fewest number of travelling technicians.

MouldMen will give us the job times and durations which they determined with MouldMen customers.

* eg. Daniel has mould and calls MouldMen. Daniel and MouldMen decide that the job will be 1:30-3:30. We then get this from the API and calculate the best allocation and travel route.

We will demonstrate Stage 1 to the client by reallocating the jobs from previous days.

We will have access to the following data:

* jobs will have:
  + only one technician required (this is for Stage 1 only)
  + duration
  + address
  + start time
* technicians will have:
  + a home address that they can start from
  + availability (usually 7am-4pm Mon-Fri)

See <https://actualisation.atlassian.net/browse/MDM-9> for more information on this data.

If a job starts before 10am, start the technician's route at their home location. If the job starts at 10am or after 10am, use the closest depot to the technician's home address for the start of their route. The depots are listed below:

* 4/37 Expansion St, Molendinar QLD 4214
* G, 135-153 New South Head Road, Edgecliff, NSW 2027
* Level 4, 80 Market St, South Melbourne, VIC 3205

If multiple technicians are the same travel time away from a job, you may choose either one.

MouldMen sometimes schedules jobs outside of a technician’s availability (eg. on the weekend). In the early stages, only schedule within a technician’s availability. If the program is told to allocate a job outside of a technician’s availability, output a warning and do not allocate that job. In later stages, after hours work must be approved by the technician prior to allocation and the work must satisfy the fatigue/work rules.

Assume that jobs will not run overtime or undertime by more than 15 minutes. You will need to create a buffer of 15 minutes after each job in which the technician can use if they go overtime.

All jobs must be completed. A day must never be scheduled which contains jobs that cannot be completed.

**4 Possible implementation:**

This implementation uses OR Tools.

**Job start time:** We can use Time Window Constraints to implement job start times. For example, if the job starts at 10am, the start window will be 9:45-10:15am. Alternatively, we can use a start window of 10:15am-10:15am (yes, that's a 0 minute window) and allow a 30 minute waiting time (this is a feature of OR Tools). Or some combination of the two.

**Job duration:** OR-Tools is designed for postal (or Amazon) deliveries, where the delivery/pickup itself is less than 1 minute (negligible compared to the travel time). However, MouldMen have jobs that might take 4 hours. To solve this, we can modify the time\_callback function to add the duration of the job to the travel time. We will also need to push back the corresponding time windows by the same amount. **This will mean OR-Tools actually schedules everything off the end time for jobs instead of the start time.**  
Note: Instead, we might be able to adapt the loading and unloading time constraints and apply it to the job instead of the depot: <https://developers.google.com/optimization/routing/cvrptw_resources#add_time_windows_for_loading_and_unloading>

def time\_callback(from\_index, to\_index):

"""Returns the travel time between the two nodes."""

# Convert from routing variable Index to time matrix NodeIndex.

from\_node = manager.IndexToNode(from\_index)

to\_node = manager.IndexToNode(to\_index)

return data["time\_matrix"][from\_node][to\_node] + job\_durations[from\_index]

# modification is underlined --> ---------------------------

**5 Change log**

This will indicate any changes in logic or new information added to the task after it is moved to “In Progress”.

There are no changes.

Next stage: <https://actualisation.atlassian.net/browse/MDM-36>