An agent-based simulation system for multi-project scheduling under uncertainty

Song, Wen; Xi, Hui; Kang, Donghun; Zhang, Jie

2018

Song, W., Xi, H., Kang, D., & Zhang, J. (2018). An agent-based simulation system for multi-project scheduling under uncertainty. *Simulation Modelling Practice and Theory*, 86, 187-203. doi:10.1016/j.simpat.2018.05.009

<https://hdl.handle.net/10356/105766>

<https://doi.org/10.1016/j.simpat.2018.05.009>

© 2018 Elsevier B.V. All rights reserved. This paper was published in *Simulation Modelling Practice and Theory* and is made available with permission of Elsevier B.V.

Downloaded on 26 Apr 2023 20:47:05 SGT

An Agent-based Simulation System for Multi-Project Scheduling under Uncertainty

Wen Song^{a,*}, Hui Xi^b, Donghun Kang^a, Jie Zhang^c

^a*Rolls-Royce@NTU Corp Lab, Nanyang Technological University, Singapore*

^b*Rolls-Royce Singapore Pte Ltd, Singapore*

^c*School of Computer Science and Engineering, Nanyang Technological University, Singapore*

Abstract

This work proposes an agent-based simulation system to address challenges in resource constrained multi-project scheduling under uncertainty. The system aims at helping users develop robust schedules by taking care of intra- and inter-dependencies between projects, conflicting demands for limited resources, and the potential knock-on effect of unpredictable events. With the interactive graphical user interface embedded in the system, users are enabled to investigate project process, evaluate modified schedule, forecast impact resulted from unexpected disruptions, and conduct various what-if analysis. Reactive and proactive scheduling algorithms have been developed in the simulation system to resolve unforeseen disruptions and increase the robustness of the solutions. Experiments on a real-world case study of a technology-intensive industrial product development program have been conducted to evaluate the effectiveness of the proposed algorithms.

Keywords: Agent-based modeling, Interactive simulation platform, Dynamic project scheduling

*Corresponding author

Email addresses: songwen@ntu.edu.sg (Wen Song), hui.xi@rolls-royce.com (Hui Xi), donghun.kang@ntu.edu.sg (Donghun Kang), zhangj@ntu.edu.sg (Jie Zhang)

1. Introduction

Scheduling multiple projects that run simultaneously in a dynamic environment remains a major challenge in project management. On one hand, hard constraints must be satisfied by feasible schedules, including task precedences and resource capacities. On the other hand, projects are often executed in dynamic environments with various sources of uncertainty [1, 2], where project rescheduling is frequently required and may impose an impact on other projects due to inter-dependencies among projects. While classical mathematical approaches help deal with scheduling problems with low complexity or in a relatively static environment, it has a limitation in describing interrelationships in multi-project environments and adapting the analysis to dynamic change [3].

To this end, the agent-based approach has been identified as an effective approach for scheduling problems with high complexity [4]. More specifically, many existing studies have focused on the resource constrained multi-project scheduling problem (RCMPSP) and have proposed scheduling algorithms to appropriately allocate resources among competing projects beforehand [5, 6, 7]. However, few approaches have been developed to deal with possible on-the-fly changes, which can be resulted from various sources of uncertainty, such as unpredictable disturbances or user interference. For multi-project problems, there are usually various dependencies including project-project and project-resource relations, which make it difficult and/or inefficient for conventional scheduling algorithms to conduct frequent online rescheduling. On the other hand, Chaari et al. [4] conducted a comprehensive survey on the algorithms for scheduling and rescheduling under uncertainty and listed several prospective research directions in this domain. One promising direction is to combine multi-agent approaches with optimization techniques. While the multi-agent technology has been widely employed in dynamic scheduling and dynamic control, incorporating optimization techniques with agents' behavior logic would increase their intelligence which leads to optimal decision making for both short-term and long-term objectives.

Thus, in this paper, we propose an agent-based simulation system to help users monitor the project execution process and provide decision making support for dynamic scheduling. An interactive graphical user interface (GUI) has been integrated with the simulator to enable user-in-the-loop experimentation.

35 A real-world technology-intensive industrial product development program has been adopted as a case study to illustrate the various features provided by our simulation system. This paper extends our previous work [8] with full details of the simulation system, including the model development and system implementation. In addition, we design a new dynamic scheduler, which has the

40 functionalities of generating robust proactive schedules based on the probability knowledge about the uncertainties, and resolving schedule disruptions during simulation. Specifically, we propose a new proactive scheduling algorithm to deal with the time-dependent workability uncertainty which is not addressed in [8], and implement a simple but effective reactive scheduling algorithm to

45 restore the feasibility of a schedule when it is disrupted.

The remainder of the paper is organized as follows: Section 2 gives a brief review on the existing studies related to our work. Section 3 introduces the dynamic multi-project scheduling problem studied in this paper. Details of the system implementation on a commercial agent-based simulation platform

50 AnyLogic[®] as well as various functions provided in the system are presented in Section 4. Section 5 provides detailed descriptions of the scheduling algorithms implemented in the system. Finally, Section 6 provides concluding remarks and proposes future research directions.

2. Related Work

55 In this section, we first give an overview of the agent-based approaches for solving multi-project scheduling problems. Then, we briefly review the approaches for project scheduling under uncertainty.

2.1. Agent-based Multi-Project Scheduling

Multi-project scheduling has gained growing attention in recent decades from both industry and academia. It concerns with the generation of resource and precedence feasible schedules to operate projects while optimizing scheduling objective(s). Multi-project scheduling is important for companies in diverse industries (e.g. software development, construction engineering, and make-to-order manufacture) as it plays a critical role in ensuring project completion with quality in a timely manner. In practice, the tasks in a project consume resources during their process, and the required resources are limited. Such kind of problem, termed as the resource-constrained multi-project scheduling problem (RCMPSP), has attracted a significant amount of research as the models in this domain are rich, complex, and difficult to solve [9]. As noticed in [3, 5], compared with traditional centralized approaches (e.g. [10, 11, 12]) assuming a single decision maker and deterministic environment, agent-based approaches are more suitable for handling large-scale real-world problems with multiple decision makers and dynamic settings.

Currently, a number of agent-based approaches for RCMPSP are available in the literature. Most of these approaches employ certain negotiation protocols to coordinate local decisions to a global solution. One of the most widely adopted protocols is auction, where a mediator agent allocates shared resources to a group of project agents based on a virtual “bidding” process. Confessore et al. [7] use a multi-agent system to solve RCMPSP, where the project managers are presented as individual agents competing for shared resources to complete their own projects. A virtual coordinator agent is developed to resolve resource conflicts via a combinatorial auction based control mechanism. In the multi-agent framework developed by Araújo et al. [3], both projects and resources are modeled as agents to deal with project portfolio management. An auctioneer agent allocates resources to projects and decides on project acceptance/rejection through the assessment of their impact on the existing portfolio. Adhau et al. [13] review some existing applications of agent-based approaches to multiple project scheduling and point out that many of them have either re-

strictive assumptions on resource types and availability [14, 15] or limitation on
90 project size (number of tasks involved) [7, 16, 17]. To address these limitations,
they propose an auction-based negotiation approach for large-scale complex
multi-project scheduling. More recently, Song et al. [18] propose a multi-unit
combinatorial auction based approach for RCMPSP, which shows significant
improvement in minimizing average project delay. Except the above mentioned
95 auction based protocols, a few approaches solve RCMPSP using other types
of mechanisms, such as the market based negotiation mechanism [19], the evo-
lutionary computation based coordination scheme [20], and the critical chains
based conflict elimination mechanism [21].

Due to the flexible structure, agent-based approaches can work in environ-
100 ments with certain types of openness and uncertainty [3]. For example, ap-
proaches in [3, 5, 19] allow projects to arrive dynamically over the planning
horizon. However, only [19] can support uncertain task durations as considered
in our problem. Nevertheless, resource capacities in [19] are assumed to be infi-
nite which contradicts the commonly adopted resource-constrained settings. In
105 the next section, we briefly review the techniques for project scheduling under
uncertainty, which mainly focus on addressing the uncertain task durations.

2.2. Project Scheduling Under Uncertainty

According to [22], approaches for project scheduling with uncertain dura-
tions can be roughly classified into two categories. The first category of ap-
110 proaches generates scheduling policies which are used to determine a feasible
schedule during execution, based on observation of the actual scenario. Differ-
ent optimization objectives have been considered, such as expected makespan
[23, 24, 25, 26] and robust makespan [27, 28, 29]. However, these approaches
do not generate a deterministic baseline schedule, which serves very important
115 functions for better preparing and coordinating the project execution [30].

In this paper, we follow the second category of approaches, which generate
a baseline schedule using a proactive procedure before execution, and repair the
baseline schedule when it is disrupted in execution using a reactive procedure. In

[31], multiple algorithms are presented to generate proactive baseline schedules
 120 by inserting time buffers between activities. In [32], several heuristics for reactive scheduling are introduced to minimize the stability cost, which measures the deviation from the actual schedule to the baseline schedule. Different combinations of proactive and reactive procedures are compared empirically in [33]. A pure proactive approach is introduced in [30] to generate baseline schedules
 125 that are feasible with a high probability during execution.

The above mentioned proactive/reactive scheduling approaches are based on the problem model where the uncertainty in activity duration is not related to the time it is scheduled, which cannot cover all the sources of uncertainty [34]. A typical example is the workability uncertainty considered in our problem, which
 130 contradicts the time-independent assumption and therefore cannot be solved by previous approaches. In Section 5, we design a sampling based proactive algorithm to address this uncertainty.

3. The Product Development Program

In this paper, we consider the dynamic scheduling of a real-world industrial
 135 product development program with high risk and complexity. In this program, multiple projects are being executed concurrently. The objective is to complete all the projects as soon as possible, i.e. to minimize the program makespan, since the delay will incur significant financial loss. Each project is assigned to a dedicated developer who will respond to disruptions occurring in the execution.
 140 A project consists of several to tens of tasks, each of which needs to be completed on one of several compatible facilities. Each task contains dozens to several hundreds of subtasks. During the task execution, all the subtasks are performed using certain instruments in every single day in parallel. Therefore, all subtasks of a task can be scheduled together as a whole. One major
 145 challenge in the product development program scheduling is to cope with complicated constraints related to both facilities and tasks which are explained in the following:

- 150

• Resource constraints. The (renewable) resources considered in this work are the facilities. Two types of resource constraints are considered:
1) compatibility: each facility is able to handle certain task types; 2) capacity: one facility can only accommodate one task at a time period, i.e. a facility can be considered as a renewable resource with unary capacity in each time slot.
- 155

• Task dependencies. Essentially, the task dependencies impose constraints on the task execution orders. Two types of dependencies exist in our product development program: 1) intra-dependencies: tasks within a project follow a specific sequence determined based on substantial domain knowledge and cannot be modified; and 2) inter-dependencies: some tasks are defined as predecessors of other tasks in different projects. For intra-dependencies, transportation and buffer times are also imposed as constraints, which will be further detailed in Section 5.1.

Our product development program is executed in a dynamic environment, where the execution of the projects in the program may be affected by uncertain events. Specifically, two types of uncertainty are involved:

- 165

• Workability uncertainty. Execution of a task requires certain weather conditions to be satisfied, e.g. temperature and precipitation. Therefore, the *workability* of a time slot (day) in the scheduling horizon for a task is uncertain. Nevertheless, a task must secure enough workable days during its scheduled execution period. If this is not satisfied at its scheduled completion time, its duration must be extended. Therefore the duration of a task is uncertain and subjects to the workability uncertainty. Due to seasonality, the Probability of Workability (POW) of each time slot (day) could be different, and in consequence, the task duration uncertainty is dependent on its scheduled start time.
- 175

• Subtask failure. The instruments may become faulty during the execution, and lead to subtask failure when the number of faulty instruments

reaches the limit. When a subtask fails, it must be rerun from the beginning, which could extend the duration of the task to which it belongs. For a type of instrument, its failure is characterized by error rate, which
180 is the same throughout the scheduling horizon and not time-dependent.

The aforementioned two types of uncertainty can both cause unexpected elongation of task durations, which could further result in schedule disruptions that must be resolved before the program execution can proceed. In the real world, resolutions are normally provided by domain experts who derive solutions by leveraging the substantial experience and situation-specific heuristics.
185 However, with growing number of projects and their complexities, manual disruption management suffers from inefficiency and ineffectiveness. For instance, if a failed task needs to be rescheduled, it may result in the following several tens or hundreds of tasks being postponed and/or losing priority on the requested
190 facilities. In addition, frequent resolution of disruptions could result in a final schedule that deviates much from the baseline schedule. To this end, we aim at developing an agent-based simulation system to take into account various resource constraints, intra- and inter-dependencies between tasks and projects so as to provide users with robust scheduling/rescheduling of the program and
195 efficient disruption management.

4. AGENT-BASED SIMULATION SYSTEM

In this section, we propose an agent-based simulation system as a decision support tool for the dynamic multi-project scheduling introduced in Section 3. Details about the modeling process and system features are illustrated in the
200 following subsections.

4.1. Agent-based Simulation Model Development

This section describes the system development in detail. Beyond the baseline of mimicking the program execution process closely, the system design emphasizes on-the-fly user interaction with the simulation via a GUI. The system

Figure 1: Proposed simulation system structure

Table 1: Attributes of different object types

Objects	Attributes
Task	name, type, required facility, duration, start time, predecessors
Subtask	name, required instruments
Facility	name, capacity for each instrument, compatible task types, available time slots

structure depicted in Figure 1 shows that input data, embedded agent behaviors, and program logic are accessible to users through the interactive GUI. In the following, we first introduce the modeling of the different components within the program.

The system employs both objects and agents in the simulation model. Tasks, subtasks, and facilities are modeled as objects as they do not perform autonomous behaviors. Table 1 lists the major attributes of each object type. On the other hand, developers and projects are modeled as agents:

- **Developer agent.** Each developer is assigned with one project. Their responsibilities include adjusting task setting, monitoring project progress, and resolving unexpected disruptions. It should be noted that developer agents within a product development program share the same goal – to complete all the projects as soon as possible. Thus, developers are fully collaborative and willing to postpone their own tasks to make room for the disrupted tasks from other projects. However, an ongoing task cannot

220 be stopped or postponed (unless disrupted by unforeseen events) until it
is completed, since the tasks are non-preemptive.

- **Project agent.** Each project agent is responsible for bookkeeping its own specifications (e.g. required facility, task sequence, and subtask list of each task) as well as updating its schedule visualization. In other words, if a
225 task has been rescheduled (e.g. postponed to another start time), relevant project agents would update their states accordingly.

Besides the above two types of agents, a Main agent is also implemented to control and coordinate the simulation. In addition, it is equipped with the dynamic scheduling algorithms for the purpose of generating baseline schedules
230 and resolving schedule disruption, which will be introduced in the next subsections. To illustrate the behaviors of the agents, we plot their UML state machine diagrams in Figure 2. Further, we plot the UML sequence diagram in Figure 3, which details the message exchanging among agents during the simulation process.

235 4.2. Dynamic Scheduling Algorithms

To guarantee the correct execution of simulation, scheduling algorithms are needed to generate or repair schedules of the product development program. Specifically, two algorithms are required during simulation:

- **Proactive scheduling algorithm.** Before the start of simulation, a
240 baseline schedule should be generated, which provides the facility assignments and start times of all tasks. Any feasible schedule can serve as the baseline, such as the one generated by solving a deterministic problem without considering any uncertainty. However, this deterministic schedule could be disrupted frequently, resulting in intensive efforts for disruption
245 resolution and final schedules that deviate too much from the baseline. Therefore, a better alternative is to develop *proactive* algorithms that exploit the stochastic knowledge about the uncertainty, which can provide

Figure 2: The UML state machine diagrams

Figure 3: The UML sequence diagram

baseline schedules that are more robust to the uncertainty. We design such an algorithm in Section 5.3.

- 250 • **Reactive scheduling algorithm.** Though baseline schedules generated by the proactive algorithm are more robust than the deterministic schedules, unforeseen disruptions cannot be fully avoided during simulation. Therefore, *reactive* algorithm is needed whenever the baseline schedule is disrupted. One way to resolve the disruption is to reschedule all the
255 remaining tasks that have not started yet using the proactive algorithm. However, this could impose major modifications to the baseline schedule which are undesirable. Therefore, we implement a simple reactive algorithm in Section 5.2 which keeps the facility assignments in the baseline schedule unchanged, and postpones the tasks affected by the disruption
260 to their earliest feasible start times.

4.3. System Implementation

The proposed simulation has been implemented in AnyLogic[®] simulation Software. As one of the most widely used agent-based modeling tool, AnyLogic’s visual development environment significantly speeds up the development process, compared to other popular agent development tools such as JADE and
265 Repast. In addition to implementing the various functionalities accordingly, we are equally motivated to develop an effective decision support system with an intuitive visual interface. AnyLogic’s simple yet sophisticated animation functionalities allow the development of visually rich, interactive simulation environments [35]. Moreover, models developed in AnyLogic can be exported as
270 a standalone Java application so that users can deploy them on other machines without the software being installed. This section explains different functionalities of the proposed system and how they work to help achieve robust program scheduling. In the following subsections, we will introduce the GUIs implemented in the simulation system, including parameter setting, interactive GUI
275 for simulation and user-in-the-loop experiment.

Product Development Program Simulation

Simulation Setting

External Database

Task Configuration

Task Schedule:

Task Duration:

Task Type:

Task Complexity:

SubTask Spec:

Buffer Time: 1 days

Minimum Instrument Demand Required (%): 60

Facility Constraints

Instrument Capacity:

Type Compatibility:

Time Availability:

Transport Time:

Task Dependency:

Instrument Error Rate

Instrument1	<input type="range"/> 0.05	Instrument9	<input type="range"/> 0.05
Instrument2	<input type="range"/> 0.05	Instrument10	<input type="range"/> 0.05
Instrument3	<input type="range"/> 0.05	Instrument11	<input type="range"/> 0.05
Instrument4	<input type="range"/> 0.05	Instrument12	<input type="range"/> 0.05
Instrument5	<input type="range"/> 0.05	Instrument13	<input type="range"/> 0.05
Instrument6	<input type="range"/> 0.05	Instrument14	<input type="range"/> 0.05
Instrument7	<input type="range"/> 0.05	Instrument15	<input type="range"/> 0.05
Instrument8	<input type="range"/> 0.05		

☐ Define failures in tasks

Figure 4: Parameter setting

Figure 5: Interactive Gantt chart

4.3.1. Parameter Setting

Input data can be loaded in two ways: linking to external database or being defined by users. In the simulation setup page (see Figure 4), users can import
280 relevant data stored in external databases. For some parameters of particular interests, while default values are given, users can also adjust their values either before or during simulation run. For instance, in the product development program, users are keen to find out how the instrument reliability can impact the program. They can vary the reliability of certain instruments at the simulation
285 setup page. The system also provides on-the-fly parameter change so that users can experiment with different values without restarting the simulation.

4.3.2. Interactive Graphical User Interface

Figure 5 is a snapshot of a running simulation as viewed in the interactive GUI. The interface design is based on the Gantt chart so as to minimize
290 users' learning curve from conventional program management tools such as Microsoft Project. In addition to making basic project information available in normal Gantt charts (e.g. start date, task length, work breakdown structure, and precedence relationships), our enhanced design also displays task-facility relations and schedule deviation to the current time. As shown at the bottom of
295 the GUI, each facility is designated with a particular color which matches with that of tasks requesting it. A vertical line in blue color representing "Today" moves with simulation progress to give an overview of the current program status. The color of each task denotes its requested facility. Two types of conflicts are represented with designated colors as well.

300 As shown in the GUI, several options (next to the play button) are available for users to interrupt the running simulation. Other than the "Change Parameter Value" option mentioned in Section 4.3.1, users can obtain detailed information about individual tasks by selecting "Check Task Status" and clicking on any task of their interest. Figure 6 shows information about an ongoing
305 task Task3-1 including its subtask list, number of failed instruments of each subtask, and percentage of the task being completed.

Figure 6: Check task status

Figure 7: Define failures in future tasks

4.3.3. *User-in-the-Loop Experimentation*

It is common for many simulation applications to have the human users in the loop, meaning that the human users can interact with the simulator in real time to conduct operations that can affect the results of simulation [36, 37]. This functionality enables the users to actively participate in the simulation process in many ways, for example performing experiments on interesting scenarios, or reacting to the outputs of other simulating agents. In our simulation system, besides the on-the-fly parameter change as explained in Section 4.3.1, user-in-the-loop experimentation is also enabled in the following two ways.

Firstly, users can define subtask failures in future tasks before or during simulation. This function is useful when users want to evaluate how the program will be affected when certain tasks fail on specific days. Note that different from the failures caused by the random faulty instruments, the user-defined subtask failures will certainly happen at the times specified by users. By clicking on an unfinished task, a list of subtasks will pop up for users to choose any subtasks to be failed before the task completes. Users will be notified with a feasible day range whenever each selected subtask fails as specified (see Figure 7). Users can also view the defined failures and cancel the ones that have not occurred yet (see Figure 8).

Secondly, at any point of a simulation run, users can manually adjust schedules for tasks that have not started yet. This function could be used to examine the impact of different task start times. Users can adjust a project schedule by dragging and dropping the tasks. The system will automatically verify if the adjustment results in a feasible schedule, according to the constraints mentioned in Section 3. For instance, users will not be able to move a task prior to its predecessors. The moving task will be marked with grey color if its schedule overlaps that of others. Once users are done with the manual adjustment, the system will check if there is any newly introduced resource conflicts before resuming the simulation run with the adjusted schedule.

Figure 8: View/edit defined failures

5. Dynamic Scheduling Algorithms

In this section, we introduce the algorithms we designed to cope with the uncertainties in the scheduling problem studied in this paper. We first describe the deterministic scheduling problem which does not consider uncertainty, along with a solution algorithm. Next, we introduce a simple algorithm for resolving schedule disruptions caused by unforeseen events. Finally, we design a novel algorithm for generating robust proactive schedules that are adapted to the workability uncertainty.

5.1. Deterministic Scheduling

Here we describe the scheduling problem introduced in Section 3 using notations, without considering the uncertainties first. In this multi-project scheduling problem, a set of projects $\mathbf{P} = \{P_1, \dots, P_I\}$ in the program need to be scheduled onto a set of facilities $\mathbf{F} = \{F_1, \dots, F_K\}$ in a horizon of T consecutive time slots (days). Each project P_i , $1 \leq i \leq I$ consists of a set of non-preemptive tasks $\mathbf{A}_i = \{a_{i1}, \dots, a_{iJ_i}\}$, where J_i is the number of tasks in P_i . We denote $\mathbf{A} = \cup_{i=1}^I \mathbf{A}_i$ as the set of all tasks in the program, and index the elements in $\mathbf{A}$ by l , $1 \leq l \leq L = \sum_{i=1}^I J_i$. Each task $a_l \in \mathbf{A}$ must be processed for d_l time slots, which is a fixed value called the baseline duration. In addition, each a_l can only be processed by a compatible facility $F_k \in \mathbf{F}_l \subseteq \mathbf{F}$, where $\mathbf{F}_l$ is the set of

compatible facilities of a_l . As stated in Section 3, all subtasks of a task should start and end together on the same facility, therefore we do not consider subtask in the scheduling problem. A solution $O = \langle M, S \rangle$ to the problem consists of two parts, namely facility assignment M and start-time schedule S . Both M and S are vectors, i.e. $M = [M_1, \dots, M_L]$ and $S = [S_1, \dots, S_L]$, where $M_l \in \mathbf{F}$ and $S_l \in \{1, \dots, T\}$ are the facility assignment and start time of task a_l , respectively. For convenience, given a solution O , we denote $s(a_l, O)$, $c(a_l, O)$ and $m(a_l, O)$ as the start time, completion time and facility assignment of $a_l \in \mathbf{A}$ specified by O , respectively. It should be noted that for a deterministic problem, once the start time of a task a_l is determined, its completion time is also determined, i.e. $c(a_l, O) = s(a_l, O) + d_l$ since the baseline duration is fixed.

A feasible solution O of the problem must satisfy the resource constraints and task dependencies, as we mentioned in Section 3. To satisfy resource compatibility, the assigned facility of task a_l must be chosen from the compatible ones, i.e. $\forall a_l \in \mathbf{A}, m(a_l, O) \in \mathbf{F}_l$. To satisfy resource capacities, since all facilities have unary capacities, any two tasks assigned to the same facility cannot overlap, i.e. $\forall a_e, a_f \in \mathbf{A}$, if $m(a_e, O) = m(a_f, O)$, then $c(a_e, O) \leq s(a_f, O)$ or $c(a_f, O) \leq s(a_e, O)$ must hold. For task dependencies, if two tasks a_e and a_f in $\mathbf{A}$ has a precedence relation $a_e \prec a_f$ which means a_f must start after the completion of a_e , then $c(a_e, O) \leq s(a_f, O)$ must hold. In addition, if a_e and a_f belong to the same project, then transportation and buffer time should be considered. To be more specific, if a_e and a_f are allocated to the same facility, i.e. $m(a_e, O) = m(a_f, O)$, then $c(a_e, O) + t_B \leq s(a_f, O)$ must hold, where t_B is the buffer time; otherwise, $c(a_e, O) + t_{T,uv} \leq s(a_f, O)$ is required, with $t_{T,uv}$ being the transportation time from facility $u = m(a_e, O)$ to $v = m(a_f, O)$. We use $\mathbf{E} = \{(a_e, a_f) | a_e, a_f \in \mathbf{A}, a_e \prec a_f\}$ to denote the set of all (intra- and inter-) dependencies between tasks in the program. In addition, let $\mathbf{Suc}_l = \{a_e \in \mathbf{A} | (a_l, a_e) \in \mathbf{E}\}$ and $\mathbf{Pre}_l = \{a_e \in \mathbf{A} | (a_e, a_l) \in \mathbf{E}\}$ be the set of immediate successors and predecessors of a_l , respectively. A solution is said to be optimal, if it minimizes the program makepan $MS(O) = \max_{a_l \in \mathbf{A}} c(a_l, O)$.

Due to the existence of multiple compatible facilities for a given task, our

scheduling problem can be considered as a special case of the multi-mode RCMPSP [38], where the duration of an task on any compatible facility is the same. Here we implement a priority-rule based algorithm based on the serial schedule generation scheme [39]. As shown in Algorithm 1, a task set $\mathbf{CS}$ is maintained during
390 execution to record the tasks that have been scheduled. An iterative scheduling procedure is conducted, where a set $\mathbf{ES}$ containing unscheduled tasks whose immediate predecessors have been scheduled is identified first (Line 3). Then, a task a_e is chosen for scheduling according to the minimum Latest Finish Time (LFT) rule [39] (Line 4), followed by the facility chosen steps which chooses
395 a facility F_k from the compatible set $\mathbf{F}_e$, according to the minimum Earliest Feasible Finish Time (EFFT) rule proposed in [40] (Lines 6-8). Specifically, the LFT value of each task is obtained before executing Algorithm 1 by applying the critical path analysis [39], while the EFFT value of a task on a facility is computed during Algorithm 1, by finding the earliest feasible start time to
400 schedule the chosen task (Lines 6-7). The transportation and buffer times are considered in computing the earliest start times. Finally, the chosen task a_e is scheduled to start at its earliest feasible start time on the chosen facility (Line 9), and is incorporated in $\mathbf{CS}$ (Line 10). This algorithm has a complexity of $\mathcal{O}(L^2K)$.

405 5.2. Reactive Scheduling

Due to the uncertainties in Section 3, the actual duration of task a_l could be longer than its baseline duration d_l . When this happens, a feasible solution could be disrupted since the resource constraints and task dependencies could be violated. In this case, a disruption resolving step, i.e. a reactive scheduling
410 procedure, is needed to restore the feasibility of the solution. Since the solution disruptions could happen quite often during the simulation, the reactive scheduling procedure should repair the solution rapidly, otherwise the time for the simulation could become prohibitive.

Currently, we have implemented a simple but efficient postpone strategy for
415 reactive scheduling, as shown in Algorithm 2. Whenever a solution disruption

Algorithm 1: Deterministic Scheduling Algorithm

Input: A deterministic scheduling problem

Output: O : a feasible solution

```
1  $CS \leftarrow \emptyset$ ;
2 while  $|CS| \leq L$  do
3    $ES \leftarrow \{a_l \in A - CS | Pre_l \subseteq CS\}$  ;
4   Select  $a_e \in ES$  with the minimum LFT ;
5   foreach  $F_k \in F_e$  do
6     Compute the smallest time  $s^k$  such that  $a_e$  can be scheduled in the
       interval  $[s^k, s^k + d_l]$  ;
7     EFFT value of  $F_k$  is  $s^k + d_l$  ;
8   Select  $F_k \in F_e$  with the minimum EFFT ;
9    $s(a_e, O) \leftarrow s^k, c(a_e, O) \leftarrow s(a_e, O) + d_e, m(a_e, O) \leftarrow F_k$  ;
10   $CS \leftarrow CS \cup \{a_e\}$ ;
11 return  $O$  ;
```

Algorithm 2: Reactive Scheduling Algorithm: Repair(O, a_f, c'_f)

Input: O : a disrupted solution; a_f : the task that causes the disruption; c'_f :
the new completion time of a_f

```
1  $\bar{c}(a_f, O) \leftarrow c'_f$  ;
2  $A_f \leftarrow \text{Postpone}(O, a_f)$  ;
3 while  $A_f \neq \emptyset$  do
4    $A'_f \leftarrow \emptyset$  ;
5   foreach  $a_e \in A_f$  do
6      $A''_f \leftarrow \text{Postpone}(O, a_e)$ ;
7      $A'_f \leftarrow A'_f \cup A''_f$  ;
8    $A_f \leftarrow A'_f$  ;
```

happens, this strategy iteratively identifies the affected tasks and postpones them to an earliest feasible start time, until the solution is repaired. The inputs of this algorithm include the disrupted solution O , along with the task a_f that causes the disruption and its new completion time c'_f . Suppose a disruptive event happens at time t , then if this event is a subtask failure, $c'_f = t + d_f$, since the task needs to be rerun from the beginning. The computation of c'_f for the workability uncertainty will be discussed in the next section. Different from the deterministic problem in Section 5.1, when uncertainty is considered, the completion time $c(a_l, O)$ of a task a_l depends on the actual scenario during execution, and cannot be fully determined by the start time. We use a vector $\bar{C} = [\bar{C}_1, \dots, \bar{C}_L]$ to record the actual task completion times, and use $\bar{c}(a_l, O)$ to denote the actual completion time of a_l . The value of $\bar{C}_l$ is set to be $C_l = s(a_l, O) + d_l$ initially for each task a_l , and is kept updated until the simulation is done.

Algorithm 3: Postpone(O, a_f)

Input: O : a disrupted solution; a_f : a postponed task;

Output: $\mathbf{A}_f$: the tasks affected by the postponement of a_f

```

1  $\mathbf{A}_f \leftarrow \emptyset, a_g \leftarrow \text{GetNextTask}(a_f)$  ;
2 foreach  $a_e \in \mathbf{Suc}_f \cup \{a_g\}$  do
3    $s_{est} \leftarrow \text{ComputeEST}(a_e)$  ;
4   if  $s_{est} > s(a_e, O)$  then
5      $s(a_e, O) \leftarrow s_{est}, \bar{c}(a_e, O) \leftarrow s_{est} + d_e$  ;
6      $\mathbf{A}_f \leftarrow \mathbf{A}_f \cup \{a_e\}$ ;
7 return  $\mathbf{A}_f$  ;
```

In Algorithm 2, firstly the actual completion time of the disruptive task a_f is set to be c'_f (Line 1). Then, in Line 2, a function Postpone as shown in Algorithm 3 is called to check the start times of a set of tasks, including its immediate successors in $\mathbf{Suc}_p$ and the task a_g that are scheduled to execute right after a_f on the facility $m(a_f, O)$. For each element a_e in the set $\mathbf{Suc}_f \cup \{a_g\}$, its earliest feasible start time s_{est} is firstly computed (Line 3 of Algorithm 3). If

s_{est} is later than its scheduled start time $s(a_e, O)$, then it is affected by a_f and will be postponed to start at s_{est} . The affected tasks will also be recorded in a set $\mathbf{A}_f$ (Line 6 of Algorithm 3). Then, for each affected task in $\mathbf{A}_f$, Algorithm 2 will call the postpone function to fix any inconsistency in the schedule (Line 6 of Algorithm 2), until no task is affected, i.e. $\mathbf{A}_f = \emptyset$.

Algorithm 2 can fix a disruptive solution very efficiently with a complexity of $\mathcal{O}(L^2)$, since a task will only be postponed once. In the future, we seek to incorporate more sophisticated algorithms for reactive scheduling, e.g. the heuristics introduced in [32]. A major challenge is to adapt them to our case, which is a more complex multi-mode RCMPSP.

5.3. Proactive Scheduling

Different from the instrument failure rate which is the same on each time slot, the workability uncertainty has a seasonal pattern which could be utilized to generate robust proactive solutions. As mentioned in Section 2.2, currently there is no proactive approach that can be applied to time-dependent duration uncertainty, which is the case in our scheduling problem. In this section, we design a novel algorithm for generating proactive schedules by exploiting the probability knowledge about the workability uncertainty, based on the Consensus approach from stochastic optimization [41].

5.3.1. Workability Uncertainty Model

Under workability uncertainty, whether a time slot can be used for processing a task is uncertain. Without loss of generality, we classify all tasks in $\mathbf{A}$ into Z types according to the impact of workability uncertainty. Denote the type of $a_l \in \mathbf{A}$ as $z_l \in \{1, \dots, Z\}$. We use a vector $p_z = [p_{z1}, \dots, p_{zT}]$ to represent the Probability of Workability (POW) of task type z , i.e. tasks belonging to type z can be processed in time slot t with a probability of p_{zt} . We call the collection $P = (p_1, \dots, p_Z)$ as the model of workability uncertainty. We assume that the scheduling horizon $\{1, \dots, T\}$ is split into a set of periods $\mathbf{E} = \{E_1, \dots, E_R\}$, where $E_r = \{es_r, \dots, ed_r\}$, $1 \leq r \leq R$ is a consecutive time period (e.g. week,

Figure 9: Monthly probability of workability

season, month) that starts at es_r and ends at ed_r . We further assume that for a task type, the POW values are the same for each slot in a period. Figure 9 shows an example of the POW of four types of tasks, where the periods are the months in a year.

Under workability uncertainty, the duration of task a_l is no longer fixed as the baseline duration d_l ; instead, it is a random variable that depends on the scheduled task start time. Denote an actual scenario of workability as a matrix $Q = [q_{zt}]_{Z \times T}$, where $q_{zt} \in \{0, 1\}$ and time slot t is workable for type z if $q_{zt} = 1$. Then if a task a_l is scheduled to start at $s(a_l, O)$ by a solution O , its actual completion time should be $\bar{c}(a_l, O) = \min\{c \geq s(a_l, O) \mid \sum_{\tau=s(a_l, O)}^{c-1} q_{z_l \tau} \geq d_l\}$. In other words, a task is completed once it secures enough workable time slots specified by the baseline duration d_l .¹ If a_l is scheduled to complete at t but $\sum_{\tau=s(a_l, O)}^{t-1} q_{z_l \tau} < d_l$, then a disruptive event is triggered at t and the new completion time of a_l is set to $c'_l = t + d_l - \sum_{\tau=s(a_l, O)}^{t-1} q_{z_l \tau}$, and Algorithm 2 is called to repair any solution disruption.

¹Note that $\bar{c}(a_l, O)$ is the earliest possible completion time. Here we use this value as the task completion time since our objective is to minimize the makespan.

480 5.3.2. Consensus based Proactive Scheduling

Our proactive scheduling approach is based on the Consensus method [41], which is a sampling based approach for online stochastic optimization problem. It first samples a set of scenarios from the probability distribution, then finds a feasible solution for each scenario, and finally makes decisions through a consensus voting process by all the scenario solutions. Consensus based approaches
485 have been applied for solving various online scheduling problems [42, 43]; in contrast, here we adapt Consensus to generate offline proactive solutions that are expected to be robust under the time-dependent workability uncertainty. Our approach takes input a feasible solution generated by solving the deterministic problem using Algorithm 1, and conducts a series of consensus solving and
490 voting processes for each time period to make the scheduling decisions.

Next, we describe the proactive algorithm in detail. As shown in Algorithm 4, firstly a set of N scenarios $\mathbf{Q} = \{Q_1, \dots, Q_N\}$ is generated by sampling the workability uncertainty model P . Also, a task set $\mathbf{CS}$ is created to record
495 the scheduled tasks during the process. Then, for each period E_r , a consensus process is conducted. The two vectors V and D in Line 3 are used to record the total votes and total durations of each task. For each scenario Q_n , a feasible solution O_n is obtained by simulating the initial solution O on Q_n . Specifically, Algorithm 2 will be called during the simulation to resolve any disruptions. Then, in Lines 6-9, for each task a_l that is not scheduled, its actual duration
500 in O_n will be accumulated to the corresponding value D_l in the record vector, and its vote V_l will increase by one if it is scheduled to start within the current period E_r in O_n . Next, in Lines 10-18 a series of scheduling decisions are made in a while loop, according to the consensus vote results. In this loop, a task set
505 $\mathbf{ES}$ of the eligible tasks is maintained to incorporate the ones whose immediate predecessors have all been scheduled. Then, a function `NextTask` is called to return the task $a_g \in \mathbf{ES}$ with the maximum vote $V_g > 0$. If there is no such task, the loop is stopped and the algorithm moves to the next period. Otherwise, the estimated duration $estD_g$ of a_g is set to the rounded average duration of

Algorithm 4: Proactive Scheduling Algorithm

Input: O : a feasible solution

```

1 Initialization:  $\mathbf{Q} \leftarrow \{Q_1, \dots, Q_N\}, \mathbf{CS} \leftarrow \emptyset$  ;
2 foreach  $E_r \in \mathbf{E}$  do
3    $V \leftarrow [0, \dots, 0], D \leftarrow [0, \dots, 0]$  ;
4   foreach  $Q_n \in \mathbf{Q}$  do
5      $O_n \leftarrow \text{Simulate}(O, Q_n)$ ;
6     foreach  $a_l \notin \mathbf{CS}$  do
7        $D_l \leftarrow D_l + \bar{c}(a_l, O_n) - s(a_l, O_n)$  ;
8       if  $es_r \leq s(a_l, O_n) < ed_r$  then
9          $V_l \leftarrow V_l + 1$  ;
10   $\mathbf{ES} \leftarrow \{a_l \notin \mathbf{CS} | Pre_l \subseteq \mathbf{CS}\}$ ;
11   $a_g \leftarrow \text{NextTask}(V, \mathbf{ES})$  ;
12  while  $a_g \neq \text{null}$  do
13     $estD_g \leftarrow \lceil D_g / N \rceil$  ;
14     $s(a_g, O) \leftarrow \text{ComputeST}(a_g, estD_g, m(a_g, O))$  ;
15     $\text{Repair}(O, a_g, s(a_g, O) + estD_g)$ ;
16     $\mathbf{CS} \leftarrow \mathbf{CS} \cup \{a_g\}$  ;
17     $\mathbf{ES} \leftarrow \{a_l \notin \mathbf{CS} | Pre_l \subseteq \mathbf{CS}\}$ ;
18     $a_g \leftarrow \text{NextTask}(V, \mathbf{ES})$  ;
19  if  $|\mathbf{CS}| = L$  then
20    break;

```

510 a_g in all scenarios (Line 15). Then, the start time of a_g will be set to the earliest precedence and resource feasible start time computed by the function ComputeST with the estimated duration $estD_g$ (Line 16), followed by a step of calling Algorithm 2 to resolve any disruptions in O (Line 17). The algorithm will terminate if all tasks have been scheduled (Lines 19-20), and finally O 515 has become a proactive solution that is adapted to the workability uncertainty model. The most time consuming step in Algorithm 4 is the simulation of each scenario in Line 5, therefore Algorithm 4 has a polynomial complexity of $\mathcal{O}(RNL^2KT)$.

6. Computational Results

520 In this section, we conduct experiments to verify the effectiveness of our proactive scheduling algorithm. We only consider the workability uncertainty in the experiments. The test program consists of 5 projects, each with 4-7 tasks, to be scheduled on 6 facilities. The tasks within a project should be processed in sequence, indicating a set of precedence relations (e.g. $T1 \prec T2$, $T2 \prec T3$, etc.). 525 All tasks are classified into four types, and the corresponding POW data is shown in Figure 9. The details of the test program are shown in Table 2. To generate a scenario Q from this dataset, a random sampling process is conducted for each task type on each time slot to determine the corresponding q_{zt} value. The scenarios used for both the proactive algorithm and solution evaluation will 530 be generated in this way.

We first show the impact of N , i.e. the number of scenarios used by the proactive algorithm. Intuitively, more scenarios would produce solutions with higher quality, but require longer computation time. To empirically examine the impact of sample size, we choose $N \in \{10, \dots, 100\}$ and run the algorithm 535 100 times for each N value. We plot the average makespan and computation time against N in Figure 10. As shown in this figure, with the increase of N , the makespan becomes stable while the computation time increases, which clearly shows a trade-off between the solution quality and computational efficiency. In

Table 2: The test program

Prj	Task	Dur	Type	Facility	Prj	Task	Dur	Type	Facility
P1	T1	30	4	{1,2}	P4	T1	30	2	{1,2,3,4,5,6}
	T2	30	2	{1,2,3,4,5,6}		T2	60	4	{1,2}
	T3	70	2	{1,2,3,4,6}		T3	5	2	{1,2,3,4,5,6}
	T4	10	2	{2,6}		T4	100	2	{1,2,3,5,6}
	T5	90	3	{1,2}		T5	90	4	{1,2,3,4,5,6}
	T6	90	2	{1,2,3,5,6}		T6	75	2	{1,2,3,6}
P2	T1	60	4	{1,2,3,4,5,6}	P5	T7	170	3	{1,2,3,4,6}
	T2	15	4	{1,2}		T1	25	3	{1,2}
	T3	60	2	{2,6}		T2	30	3	{1,2,3,5,6}
	T4	10	2	{1,2,3,4,5,6}		T3	50	2	{2,6}
	T5	90	2	{3,6}		T4	30	1	{3,6}
	T6	90	1	{1,2,3,4,6}		T5	70	2	{1,2,3,4,5,6}
P3	T1	45	4	{1,2,3,4,5,6}		T6	70	2	{1,2,3,4,6}
	T2	30	1	{1,2,3,4,6}		T7	60	2	{2,6}
	T3	30	2	{1,2,3,4,5,6}					
	T4	30	4	{1,2}					

the following experiments we set $N = 50$, since the solution is stable when
540 $N \geq 50$.

Next, we evaluate the optimality of the proactive solution produced by our algorithm. Specifically, we randomly generate 20 actual scenarios by sampling the workability uncertainty model. For each of these scenarios, we simulate a proactive solution O_{pro} generated by our approach to obtain an actual solution
545 $\bar{O}_{pro}$. During simulation, Algorithm 2 is used for resolving disruptions. We compare the makespan of the actual solution $\bar{O}_{pro}$ with that of the offline optimal solution O^* , which is obtained by solving a Mixed Integer Quadratically Constrained Program (MIQCP) formulated by assuming the actual scenario is fully known. This offline optimal solution is the best that can be achieved by any
550 approach. Details of the MIQCP model will be introduced in the Appendix. For each of the 20 scenarios, we generate 100 proactive solutions using our approach, therefore 100 actual solutions can be obtained. Table 3 shows the makespan of the offline optimal solution, the average makespan of the 100 actual solution,

Figure 10: Impact of sample size

Table 3: Comparison of the makespans of the offline optimal solutions and actual solutions obtained from the proactive solutions

Scenario No.	1	2	3	4	5	6	7	8	9	10
Offline optimal	880	835	841	838	877	870	853	862	891	837
Our approach	940.67	917.7	910.93	916.7	946.77	935.03	922.5	936.33	931.13	915.77
Gap(%)	6.89	9.9	8.32	9.39	7.96	7.48	8.15	8.62	4.5	9.41
Scenario No.	11	12	13	14	15	16	17	18	19	20
Offline optimal	909	866	894	865	861	848	867	859	849	890
Our approach	934.4	936.87	943.83	936.1	924.77	916	943.03	927.87	916.9	941.23
Gap(%)	2.79	8.18	5.57	8.22	7.41	8.02	8.77	8.02	8	5.76

and the average optimality gap. As shown in this table, the proactive solutions
555 generated by our approach can lead to near-optimal actual solutions, within a
10% gap to the offline optimal solution.

Finally, we evaluate the robustness of the proactive solution produced by
our algorithm. Intuitively, a baseline solution is more robust if the sensitivity of
the planned task start times to the schedule disruptions is lower [33]. In other
560 words, the actual solution should differ from the baseline solution as little as
possible. To this end, we use the stability cost [33] to evaluate the robustness of
a baseline solution, which is a widely used criterion for evaluating the solution
robustness. For a baseline solution O , its stability cost is computed as $SC(O) =$

Figure 11: Comparison of stability cost

$\sum_{i=1}^L |s(a_i, O) - s(a_i, \bar{O})|$, where $\bar{O}$ is the actual solution obtained by executing
565 O . By definition, the stability cost measures the absolute deviation of the task
start times in the baseline solution and actual solution. We compare the stability
cost of using two types of baseline solutions: 1) O_{dtm} which is the deterministic
solution generated by Algorithm 1 without considering the uncertainty model,
and 2) O_{pro} which is the proactive solutions generated by Algorithm 4. We
570 run experiments on the 20 scenarios generated above, and compare the stability
costs of the two types of baseline solutions. Algorithm 2 is used for resolving
disruptions. For each of the 20 scenarios, we also run Algorithm 4 100 times
to generate 100 proactive solutions O_{pro} , and plot the average stability cost of
these proactive solutions in Figure 11 (the vertical axis is in log scale), along
575 with that of using the deterministic solutions O_{dtm} . As shown in this figure,
the stability cost of using the proactive solutions is much lower than that of
using the deterministic solutions. This observation indicates that the proactive
scheduling procedures in Algorithm 4 can generate the baseline solutions that
are well adapted to the workability uncertainty model, therefore significantly
580 better robustness can be achieved by the proactive scheduling algorithm.

7. Conclusion and Future Work

In this work, we have developed an interactive agent-based simulation system to facilitate multi-project scheduling and decision making under uncertainty. In our simulation system implemented in AnyLogic[®], target users are enabled to
585 conduct user-in-the-loop experimentation including making on-the-fly parameter change, defining specific task failures, and adjusting future task schedule, through an interactive GUI.

A unique feature of our simulation system is the existence of workability uncertainty, which can cause the task duration uncertainty to depend on
590 its start time. Since existing approaches cannot handle such kind of time-dependent duration uncertainty, we design a novel sampling-based proactive scheduling algorithm to generate robust baseline solutions, based on the Consensus method from online stochastic optimization. We also design a rapid reactive scheduling algorithm to repair disrupted solutions. Experiments on
595 a real-world technology-intensive industrial product development program has been used to show the effectiveness of our proactive scheduling algorithm.

We are exploring several directions for future research based on the current simulation system. Firstly, we aim at adapting more sophisticated reactive algorithms in the literature to better resolve the disruptions. Secondly, more
600 complexities can be incorporated in the problem model. For instance, more complex temporal relations could exist between activities, in addition to the precedence constraints. There could also be more complex task-resource relations, e.g. the completion of a task requires cooperation of multiple facilities. We will incorporate these practical features in the future, which can further
605 improve the usefulness of our simulation system.

Acknowledgments

This work was conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the CorpLab@University Scheme.

610 References

- [1] J. T. Lin, C.-M. Chen, Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing, *Simulation Modelling Practice and Theory* 51 (2015) 100–114.
- [2] K. Nagasawa, Y. Ikeda, T. Irohara, Robust flow shop scheduling with random processing times for reduction of peak power consumption, *Simulation Modelling Practice and Theory* 59 (2015) 102–113.
- [3] J. A. Araúzo, J. Pajares, A. Lopez-Paredes, Simulating the dynamic scheduling of project portfolios, *Simulation Modelling Practice and Theory* 18 (10) (2010) 1428–1441.
- [4] T. Chaari, S. Chaabane, N. Aissani, D. Trentesaux, Scheduling under uncertainty: Survey and research directions, in: *International Conference on Advanced Logistics and Transport (ICALT)*, IEEE, 2014, pp. 229–234.
- [5] S. Adhau, M. Mittal, A. Mittal, A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach, *Engineering Applications of Artificial Intelligence* 25 (8) (2012) 1738–1751.
- [6] C.-N. Bodea, I. R. Badea, A. Purnus, Complex project scheduling using multi-agent methods: A case study for research projects, *Management & Marketing* 5 (3) (2010) 21.
- [7] G. Confessore, S. Giordani, S. Rismondo, A market-based multi-agent system model for decentralized multi-project scheduling, *Annals of Operations Research* 150 (1) (2007) 115–135.
- [8] H. Xi, C. K. Goh, P. S. Dutta, M. Sha, J. Zhang, An agent-based simulation system for dynamic project scheduling and online disruption resolving, in: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1759–1760.

- [9] L. Özdamar, G. Ulusoy, A survey on the resource-constrained project scheduling problem, *IIE transactions* 27 (5) (1995) 574–586.
- [10] I. Kurtulus, E. Davis, Multi-project scheduling: Categorization of heuristic rules performance, *Management Science* 28 (2) (1982) 161–172.
- [11] A. Lova, P. Tormos, Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling, *Annals of Operations Research* 102 (1) (2001) 263–286.
- [12] J. F. Gonçalves, J. J. Mendes, M. G. Resende, A genetic algorithm for the resource constrained multi-project scheduling problem, *European Journal of Operational Research* 189 (3) (2008) 1171–1190.
- [13] S. Adhau, M. Mittal, A. Mittal, A multi-agent system for decentralized multi-project scheduling with resource transfers, *International Journal of Production Economics* 146 (2) (2013) 646–661.
- [14] K. Kim, B. C. Paulson Jr, Agent-based compensatory negotiation methodology to facilitate distributed coordination of project schedule changes, *Journal of Computing in Civil Engineering* 17 (1) (2003) 10–18.
- [15] Y.-H. Lee, S. R. Kumara, K. Chatterjee, Multiagent based dynamic resource scheduling for distributed multiple projects using a market mechanism, *Journal of Intelligent Manufacturing* 14 (5) (2003) 471–484.
- [16] J. Lau, G. Huang, K. Mak, L. Liang, Distributed project scheduling with information sharing in supply chains: part ii theoretical analysis and computational study, *International Journal of Production Research* 43 (23) (2005) 4899–4927.
- [17] J. S. Lau, G. Q. Huang*, K. Mak, L. Liang, Distributed project scheduling with information sharing in supply chains: part ian agent-based negotiation model, *International Journal of Production Research* 43 (22) (2005) 4813–4838.

- [18] W. Song, D. Kang, J. Zhang, H. Xi, A multi-unit combinatorial auction
665 based approach for decentralized multi-project scheduling, *Autonomous
Agents and Multi-Agent Systems* 31 (2017) 1548–1577.
- [19] X. Mao, N. Roos, A. Salden, Stable multi-project scheduling of airport
ground handling services by heterogeneous agents, in: *Proceedings of
The 8th International Conference on Autonomous Agents and Multiagent
670 Systems-Volume 1*, International Foundation for Autonomous Agents and
Multiagent Systems, 2009, pp. 537–544.
- [20] J. Homberger, A (μ, λ) -coordination mechanism for agent-based multi-
project scheduling, *OR spectrum* 34 (1) (2012) 107–132.
- [21] Z. Zheng, Z. Guo, Y. Zhu, X. Zhang, A critical chains based distributed
675 multi-project scheduling approach, *Neurocomputing* 143 (2014) 282–293.
- [22] F. Deblaere, E. Demeulemeester, W. Herroelen, Proactive policies for
the stochastic resource-constrained project scheduling problem, *European
Journal of Operational Research* 214 (2) (2011) 308–316.
- [23] G. Igelmund, F. J. Radermacher, Preselective strategies for the optimiza-
680 tion of stochastic project networks under resource constraints, *Networks*
13 (1) (1983) 1–28.
- [24] B. Ashtiani, R. Leus, M.-B. Aryanezhad, New competitive results for the
stochastic resource-constrained project scheduling problem: exploring the
benefits of pre-processing, *Journal of Scheduling* 14 (2) (2011) 157–171.
- [25] S. Creemers, Minimizing the expected makespan of a project with stochas-
685 tic activity durations under resource constraints, *Journal of Scheduling*
18 (3) (2015) 263–273.
- [26] S. Rostami, S. Creemers, R. Leus, New strategies for stochastic resource-
constrained project scheduling, *Journal of Scheduling* (2017) 1–17.

- 690 [27] J. C. Beck, N. Wilson, Proactive algorithms for job shop scheduling with probabilistic durations, *Journal of Artificial Intelligence Research* 28 (1) (2007) 183–232.
- [28] N. Fu, H. C. Lau, P. Varakantham, F. Xiao, Robust local search for solving rcpsp/max with durational uncertainty, *Journal of Artificial Intelligence*
695 *Research* 43 (2012) 43–86.
- [29] N. Fu, H. C. Lau, P. Varakantham, Robust execution strategies for project scheduling with unreliable resources and stochastic durations, *Journal of Scheduling* 18 (6) (2015) 607–622.
- [30] P. Lamas, E. Demeulemeester, A purely proactive scheduling procedure for
700 the resource-constrained project scheduling problem with stochastic activity durations, *Journal of Scheduling* 19 (4) (2016) 409–428.
- [31] S. Van de Vonder, E. Demeulemeester, W. Herroelen, Proactive heuristic procedures for robust project scheduling: An experimental analysis, *European Journal of Operational Research* 189 (3) (2008) 723–733.
- 705 [32] S. Van de Vonder, F. Ballestin, E. Demeulemeester, W. Herroelen, Heuristic procedures for reactive project scheduling, *Computers & Industrial Engineering* 52 (1) (2007) 11–28.
- [33] S. Van de Vonder, E. Demeulemeester, W. Herroelen, A classification of predictive-reactive project scheduling procedures, *Journal of Scheduling*
710 10 (3) (2007) 195–207.
- [34] M. E. Bruni, P. Beraldi, F. Guerriero, The stochastic resource-constrained project scheduling problem, in: *Handbook on Project Management and Scheduling Vol. 2*, Springer, 2015, pp. 811–835.
- [35] Anylogic simulation software overview, <http://www.anylogic.com/overview>,
715 **overview**, accessed: 2017-01-06.

- [36] D. J. Folds, Human in the loop simulation, in: *Modeling and Simulation in the Systems Engineering Life Cycle*, Springer, 2015, pp. 175–183.
- [37] J. Ma, N. Kim, L. Rothrock, Performance assessment in an interactive call center workforce simulation, *Simulation Modelling Practice and Theory* 19 (1) (2011) 227–238.
- [38] V. Van Peteghem, M. Vanhoucke, A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem, *European Journal of Operational Research* 201 (2) (2010) 409–418.
- [39] R. Kolisch, Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, *European Journal of Operational Research* 90 (2) (1996) 320–333.
- [40] A. Lova, P. Tormos, F. B. Sanchís, Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules, *Inteligencia artificial* 10 (30) (2006) 69–86.
- [41] P. V. Hentenryck, R. Bent, *Online stochastic combinatorial optimization*, The MIT Press, 2009.
- [42] P. Ströhle, E. H. Gerding, M. M. de Weerd, S. Stein, V. Robu, Online mechanism design for scheduling non-preemptive jobs under uncertain supply and demand, in: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 437–444.
- [43] S. Stein, E. Gerding, V. Robu, N. R. Jennings, A model-based online mechanism with pre-commitment and its application to electric vehicle charging, in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 669–676.

- [44] A. A. B. Pritsker, L. J. Watters, A zero-one programming approach to scheduling with limited resources, The RAND Corporation, RM-5561-PR.

745 Appendix

Here we describe the MIQCP model for solving the offline scheduling problem in Section 6. The problem we are trying to formulate here is to find the optimal solution O^* assuming that a scenario $Q = [q_{zt}]_{Z \times T}$ is fully known. Theoretically, O^* is the best one can achieve, no matter what approach is used. Hence, we use O^* in the experiments to evaluate the optimality of a proactive solution.

In deterministic project scheduling, for a task a_l , only its start time $s(a_l, O)$ need to be decided by a solution O since its duration is a fixed parameter d_l , hence $c(a_l, O) = s(a_l, O) + d_l$. However, when the workability uncertainty is considered, $c(a_l, O)$ is uncertain for a given $s(a_l, O)$. If a scenario Q is known, then to guarantee that a_l receive enough processing time, a duration constraint should be added:

$$\sum_{t=s(a_l, O)}^{c(a_l, O)} q_{z_l t} \geq d_l. \quad (1)$$

Therefore, other than the start time $s(a_l, O)$ and facility assignment $m(a_l, O)$, $c(a_l, O)$ should also be decision variables.

The key point in formulating the MIQCP is how to represent the completion time of each task. In [44], an Integer Linear Program for deterministic resource-constrained project scheduling based on the “step” variables is proposed. In this formulation, for each task a_l , a set of binary variables $x_{lt} \in \{0, 1\}$ are defined, where $x_{lt} = 1$ if and only if a_l starts at time t or before. According to the definition, for a given start time $s(a_l, O)$, $x_{lt} = 0$ for all $t < s_l$ and $x_{lt} = 1$ for all $t \geq s(a_l, O)$. Therefore, $s(a_l, O)$ can be written as:

$$s(a_l, O) = \sum_{t=1}^T t(x_{lt} - x_{l, t-1}). \quad (2)$$

In other words, $s(a_l, O)$ can be represented as a linear combination of x_{lt} . Based on this idea, for each a_l , we add another set of binary variables $y_{lt} \in \{0, 1\}$, such

that $y_{lt} = 1$ if and only if a_l ends at t or later. Then, $c(a_l, O)$ can be represented by a linear combination of y_{lt} :

$$c(a_l, O) = \sum_{t=1}^T t(y_{l,t-1} - y_{lt}). \quad (3)$$

Based on the definition, x_{lt} and y_{lt} can also be used to represent the execution status of a task. Specifically, if a_l is in execution at time t , then $x_{lt} + y_{lt} - 1 = 1$; otherwise $x_{lt} + y_{lt} - 1 = 0$.

To represent the facility assignment, we introduce a set of binary variable $h_{lk} \in \{0, 1\}$, where $h_{lk} = 1$ if and only if a_l is assigned to facility F_k . Now we are ready to build the following MIQCP model:

$$\min \quad \sum_{t=1}^T t(x_{L+1,t} - x_{L+1,t-1}) \quad (4)$$

$$\text{s.t.} \quad x_{lt} - x_{l,t-1} \leq 0, \quad \forall l, t \quad (5)$$

$$y_{l,t-1} - y_{lt} \leq 0, \quad \forall l, t \quad (6)$$

$$\sum_{t=1}^T (x_{lt} + y_{lt} - 1)q_{z_l t} \geq d_l, \quad \forall l \quad (7)$$

$$\sum_{t=1}^T t(y_{e,t-1} - y_{et}) - \sum_{t=1}^T t(x_{lt} - x_{l,t-1}) \leq 0, \quad \forall l, e \in \text{Pre}_l \quad (8)$$

$$\sum_{l=1}^L h_{lk}(x_{lt} + y_{lt} - 1) \leq 1, \quad \forall k, t \quad (9)$$

$$\sum_{k=1}^K h_{lk} = 1 \quad \forall l \quad (10)$$

$$h_{lk} = 0 \quad \forall l, k \notin \mathbf{F}_l \quad (11)$$

$$x_{lt} \in \{0, 1\}, y_{lt} \in \{0, 1\}, h_{lk} \in \{0, 1\} \quad \forall l, t \quad (12)$$

In the above formulation, Equations (4) is the objective function, which is to minimize the start time of the dummy end task, which is added as the successor of all tasks to represent the makespan of the program. Equation (5) and (6) define the step functions of the task start and completion times, respectively, along with the binary constraints in Equation (12). Equation (7) represents the duration constraints defined by Equation (1). Equation (8) represents the precedence constraints. Equation (9) represents a set of quadratic constraints, which guarantee that the unary capacity of each facility in each time slot is not

violated. Finally, Equation (10) and (11) are used to guarantee that a task is assigned to only one compatible facility.