MSc Thesis

Improving Robustness For Stochastic Parallel Machine Scheduling With Robustness Measures

Author:
Loriana Pascual
5888646

Supervisors:
Marjan van den Akker
Roel van den Broek

Department of Information and Computing Sciences

November 2022

Abstract

In this thesis we look at the stochastic parallel machine scheduling problem with precedence constraints, release dates and a deadline. To deal with uncertainty in the job processing times, it is desirable to create robust schedules. Because many machine scheduling problems are complex, they are often solved with local search methods. To include robustness into the objective of a local search approach, an efficient way to quantify the robustness of a solution is required. Since exact computation of the robustness of a schedule is complex and simulation is computationally expensive, the need for surrogate robustness measures arises. Therefore, we evaluated several existing and new proposed robustness measures on their ability to estimate the true robustness, with the aim of creating stable baseline schedules bounded by a deadline. We compared the estimates of the robustness measures with the results of a Monte Carlo simulation by reporting their Spearman's rank correlation coefficients for various notions of robustness and several processing time probability distributions. Furthermore, we implemented a local search algorithm which uses the robustness measures to evaluate the quality of the schedule. We showed the effectiveness of robustness measures as indicators of true robustness and their practical application in generating stable baseline schedules.

Acknowledgements

First of all, I would like to thank my supervisors Marjan van den Akker and Roel van den Broek for their excellent guidance, enthusiasm and time investment in the past months. I enjoyed our weekly meetings and their feedback and insights were very valuable to me. I would also like to thank Han Hoogeveen as the second reader of this thesis. The enthusiasm of Marjan van den Akker and Han Hoogeveen during their lectures was very inspiring to me and led me to develop an interest in operations research. Furthermore, I want to thank Laurens Stoop for helping me set up a computer to run the experiments. Finally, I want to express my gratitude towards my family and friends for their support and encouragement.

Contents

Chapter 1

Introduction

Scheduling is a widely researched topic, as it has many applications in real-world problems such as public transportation, production, manufacturing, logistics, healthcare and many more. Scheduling can generally be described as optimally allocating limited resources to certain tasks. A special case of scheduling is *parallel machine scheduling*, where jobs have to be scheduled for processing on one of several parallel machines. In traditional machine scheduling, it is often assumed that the environment is deterministic and that all information is known in advance. However, this is not always the case in practice. In reality, the environment is subject to uncertainty which can cause the schedule not to be carried out as planned. Therefore, it is desirable to create schedules that are *robust* in the face of uncertainty.

Many deterministic machine scheduling problems are hard to solve. Considering a stochastic environment makes the problem even more complex. Therefore, these problems are often solved with local search methods. To include robustness into the objective of a local search approach, we need an efficient way to quantify the robustness of a solution. Computing the exact value for a certain characterization of robustness is hard because the distribution of the uncertainty is not always known, and the interdependencies of the jobs makes it computationally expensive. A way to estimate the robustness of a schedule is to simulate it by taking many samples from the assumed distributions of the uncertain elements (van den Akker, van Blokland, & Hoogeveen, 2013). If a sufficient number of samples is used, this will give an accurate estimate of the true robustness. However, running many simulations is computationally expensive. If we have to evaluate many schedules to determine the most optimal one in a local search procedure, simulation can become infeasible. Consequently, we need an efficient way to estimate the true robustness of a schedule. Therefore, several *robustness measures* have been developed. Robustness measures are analytical functions that use characteristics of the schedule structure to estimate robustness. The ability of a robustness measure to correctly reflect robustness depends strongly on the definition of true robustness and on the problem. Therefore, we will evaluate and compare several robustness measures on their ability to estimate various aspects of robustness, with the goal of creating robust schedules for the stochastic parallel machine scheduling problem.

1.1 Stochastic parallel machine scheduling

We study a stochastic parallel machine scheduling problem which we define as follows. We are given n jobs that must be processed by one of m identical machines that work in parallel. In the deterministic problem, each job j has a processing time p_j. We consider a stochastic variant of this problem, where the processing times are stochastic variables that follow a probability distribution D_j with average p_j. Each job j has a release date r_j. We consider a global deadline constraint, where all jobs have to be completed before a deadline d. Additionally, there are precedence constraints that state a partial ordering in which jobs

must be executed. A precedence constraint $i \prec j$ denotes that job i must be finished before job j can start. Finding a feasible schedule can be done with the following three steps. First, each job gets assigned to a machine. Secondly, we have to order the jobs on the machines. Finally, we have to assign start times to jobs.

A solution to the first two steps can be represented as a graph. In this graph, the nodes represent the jobs and each arc correspond to a precedence relation. An arc represents either a precedence constraint that is part of the problem, or an ordering on jobs assigned to the same machine. From this graph, we can compute for each job the time window in which it should be processed. The *earliest start time* (EST) of job j is defined as the maximum of its release date r_j and the earliest time at which all its predecessors are completed. The *latest completion time* (LCT) of a job is the latest time the job can finish such that the deadline is met. From the EST we can compute the *earliest completion time* (ECT) by adding the processing time p_j. We can compute the *latest start time* (LST) by subtracting the processing time p_j from the LCT.

1.2 Robust scheduling

There appears to be no consensus about a definition for robustness in literature. In general, the robustness of a schedule can be characterized as its ability to withstand disruptions. We can distinguish two types of robustness. *Quality robustness* represents a schedules ability to prevent a significant degradation in the objective when facing uncertainty. A common objective for parallel machine scheduling is makespan minimization. The makespan is the time at which all jobs are completed. When dealing with a stochastic environment, the objective function is not deterministic. Therefore, a common objective for quality robust scheduling is minimizing the *expected* or *average* makespan. Another type of robustness is *solution robustness*. A schedule is called solution robust if the schedule itself is unlikely to degrade under uncertainty. An example of a solution robust schedule is a schedule where there is little deviation from planned start times, regardless of uncertainty.

In literature, robust scheduling is mostly about minimizing the expected makespan (quality robustness) of a schedule subject to uncertainties. In some cases, however, one might not be interested in finding the shortest schedule, but rather in finding a schedule that has a high probability to respect the planned start times (solution robustness). Having a schedule that is likely the follow the planned start times has the benefit that the execution will remain reasonably predicable, regardless of variations in job durations. This can be useful to minimize waiting times of patients in hospitals or delays in public transportation, for example.

A way to increase the solution robustness is by inserting buffers between jobs (Herroelen & Leus, 2004). Buffers can serve as a way to absorb delays so that it will not propagate further into the schedule. Enforcing buffers is achieved by assigning specific start times to jobs, instead of using an *Earliest Start Schedule* ESS. An ESS means that no explicit start times are assigned to jobs, but that the jobs are started as soon as possible. This is used for makespan minimization.

Inserting maximal buffers such that the deadline will not be exceeded might make the schedule infeasible with respect to the deadline during execution, as the processing times are not deterministic. Therefore, we also need to ensure that we have a high probability that the schedule will be completed within the deadline (quality robustness). Since inserting buffers will increase the expected makespan, we might have two conflicting objectives.

In this thesis, we aim to optimize the adherence of planned start times while meeting the deadline. Therefore, we adopt the following definitions of robustness:

Definition 1.2.1 (Quality robustness)**.** A schedule is said to be quality robust if it has a high probability of finishing within the deadline.

Definition 1.2.2 (Solution robustness)**.** A schedule is said to be solution robust if the jobs have a high probability to start at their planned start time.

1.3 Defining Slack

Many robustness measures are based on slack. Slack is idle time between jobs where the machine is not being used. Slack is closely related to the robustness of a schedule, as slack makes it possible to delay a job without causing problems. We can define two different types of slack: total slack and free slack. Total slack is closely related to the quality robustness, while free slack is closely related to the solution robustness. Common definitions of total slack and free slack in literature assume the use of an earliest start schedule. As our approach includes assigning specific start times to jobs, we have to reformulate these definitions to consider the *planned start time* (PST) instead of the earliest start time (EST). We adopt the following definitions of total slack and free slack:

Definition 1.3.1 (Total slack)**.** The total slack of a job is the maximum amount of time that we can delay a job without exceeding the deadline.

$$TS_j = LST_j - PST_j \tag{1.1}$$

Definition 1.3.2 (Free slack)**.** The free slack of a job is the amount of time by which we can delay the job without delaying any other job in the schedule.

$$FS_j = \min_{j \prec i} PST_i - PST_j - p_j \tag{1.2}$$

1.4 Research objectives

The goal of this thesis is to research the following: How can we efficiently find solution robust schedules for the stochastic parallel machine scheduling problem using robustness measures?

In order to investigate this, we will answer the following questions:

1. What robustness measures are able to make a good distinction between schedules in terms of solution robustness?
2. What robustness measures are able estimate well if a schedule will respect its deadline?
3. How efficient can these robustness measures be computed?
4. How can we divide the available slack in such a way that the solution robustness is maximal?
5. What is the effect of using these robustness measures in a local search procedure on the schedule solution and quality robustness?

Chapter 2

Literature overview

In this chapter we discuss literature about robust scheduling. We look into literature on robustness measures and on simulation studies performed to examine the quality of robustness measures. Additionally, we address literature on algorithms designed to generate stable baseline schedules to improve solution robustness.

2.1 Robustness measures

In this section, we discuss literature on several slack-based robustness measures and robustness measures that are based on the processing times probability distributions.

2.1.1 Slack-based robustness measures

Total slack

Jorge Leon, David Wu, and Storer (1994) study the robustness of job shop schedules. They define the schedule delay as the difference between the deterministic makespan and the realized makespan after execution with disruptions. The *expected* delay is then defined as the difference between the deterministic makespan and the expected makespan. The expected delay is an important measure if we want to minimize the actual makespan. However, Jorge Leon et al. argue that the expected delay by itself has little meaning, as it can be decreased easily by inserting a lot of idle time in the schedule. In contrast to the expected delay, the expected makespan also contains information about the makespan itself. The expected makespan can therefore be an important performance characteristic of a schedule. That is why Jorge Leon et al. state that a robust schedule should have a high performance regarding both the expected delay and the expected makespan. Therefore, they define the schedule robustness as a linear combination of the two.

Jorge Leon et al. develop procedures based on the graph representation of the schedule to compute the expected delay and the expected makespan assuming only *one* disruption occurs. However, these procedures are not computationally feasible if more than one disruption occurs, since the effect of disruptions depend on the outcome of all previous disruptions. Therefore, Jorge Leon et al. develop several robustness measures to approximate the true robustness, that are easy to compute.

They base a robustness measure on the assumption that the expected delay caused by the first disruption is a good indicator of the expected delay after multiple disruptions. With this assumption, they compute the robustness for the single disruption case, for each machine, and then taking the average. Another robustness measure is based on taking the integral of a function that calculates the delay in makespan caused by one disruption for a given arrival time and duration of the disruption. The robustness measure is computed by taking that integral for each machine, and then average over all machines. Jorge Leon et al.

argue that a schedule with a larger amount of slack will have less delays after a disruption. Therefore, they propose the *average of the total slacks of all jobs* as a robustness measure.

To determine the correlation between the proposed measures and the makespan and the delay after the disruptions, Jorge Leon et al. perform a simulation study on many job shop schedules. The results show that the three measures have a higher correlation with the expected delay than the deterministic makespan does. This means that their proposed measures provide additional information about the expected delay that is not captured in the deterministic makespan. They show that the average total slack is a good estimator of the expected delay. Therefore, they develop a genetic algorithm based on this measure to generate robust schedules. Experiments show that these generated schedules are less sensitive to disruptions than schedules that are made to minimize the makespan.

Hazır, Haouari, and Erel (2010) investigate several robustness measures for the discrete time/cost trade-off problem (DTCTP). These robustness measures are mainly based on total slack, as they aim to generate quality robust schedules. They perform a monte carlo simulation to asses the quality of the measures. They evaluate the effect of disruptions with two performance metrics: The fraction of samples where the deadline is met, and the average delay in the makespan as percentage of the deadline. With the simulation results they compute the correlation between the performance metrics and the robustness measures.

Hazır et al. show that their measure of the *project buffer size* has the highest correlation with both performance metrics. They define the project buffer as the time that the completion time of the project can be delayed without exceeding the deadline. The robustness measure is then defined as the project buffer as a percentage of the deadline.

Following this, they propose a two-stage algorithm for finding robust schedules. In the first phase, they assume the job durations are deterministic, and determine the minimum required budget. This budget is then used as a threshold in the next step. In the second phase, they maximize the project buffer while keeping into account the budget threshold. They show that this two-stage approach produces quality robust schedules.

Sum of free slacks

Al-Fawzan and Haouari (2005) introduce the concept of robustness for the resource-constrained project scheduling problem (RCPSP). They define the Bi-objective Resource Constrained Project Scheduling Problem (BRCPSP), in which the objective is to find a feasible schedule that minimizes the makespan while also maximizing the robustness. They define robustness as the ability of a schedule to handle (small) increases in the job processing times. They propose to measure the robustness of a schedule as the *sum of the free slack of all jobs*.

Using this robustness measure, Al-Fawzan and Haouari design a multi-objective tabu search algorithm. Since the problem has two objectives, there may be multiple efficient solutions. That is why they propose an adaptation of tabu search to find an *approximate set* of non-dominated schedules. A schedule is called non-dominated (efficient) if it is not possible to improve *all* objectives without violating constraints.

In their tabu search procedure, they compute the makespan and the robustness for each schedule in the neighbourhood. They select the new current solution based on an aggregation function that combines the values for the makespan and the robustness. The aggregation function uses an importance weight to set a balance between makespan minimization and robustness maximization by making a linear combination of the two objectives. Al-Fawzan and Haouari observe that the robustness value is often much larger than the makespan value, which causes the robustness to strongly dominate the makespan. To avoid this, they define a variant of the aggregation function, that only considers the relative improvement of the makespan and the relative improvement of the robustness.

Al-Fawzan and Haouari perform experiments with several variants of their algorithm. They show that if the best variant of their algorithm (which is based on the relative improvement of the makespan and the robustness) is used to minimize the makespan only, it performs comparable to tabu search algorithms specifically designed for single objective

makespan minimization.

Minimum free slack

Kobylański and Kuchta (2007) critique that Al-Fawzan and Haouari do not research if their proposed robustness measure of the sum of free slacks is actually related to the true robustness. Kobylański and Kuchta show with an example that maximizing the sum of free slacks can result in schedules that are absolutely not robust. Therefore, Kobylański and Kuchta propose two other robustness measures.

They argue that maximizing the *minimum of free slacks* will protect the solution robustness as well as the quality robustness and they show that this measure is better than the sum of free slacks as proposed by Al-Fawzan and Haouari. The second robustness measure is based on the idea that in many cases, the longer the processing time of a job, the higher the probability of a longer delay. Therefore, they propose to use the *minimum of the ratios free slack/processing time* as a robustness measure. They show that this measure is better then the sum of free slacks and can be better than their first robustness measures in some cases. Additionally, Kobylański and Kuchta argue that for practical applications, it is better to fix the makespan and determine a robust schedule for that makespan, then it is to use weights to give importance to the makespan and the robustness.

Weighted free slack

To avoid the dilemma of choosing between minimizing the makespan and maximizing quality robustness, Chtourou and Haouari (2008) develop a two-stage-priority-rule-based approach for the RCPSP. They first solve the problem for minimizing the makespan. From the found schedules with a small makespan, they then maximize the quality robustness.

To estimate the robustness of a schedule, they use 12 robustness measures that are based on the sum of free slacks, as defined by Al-Fawzan and Haouari (2005). Chtourou and Haouari argue that an increase in the processing time of a job that has a large number of successors is more likely to affect the makespan than when a job has only a few successors. That is why they propose the sum of the free slacks weighted by the number of successors as a robustness measure. Furthermore, they indicate that increase in the processing time of a job that requires a lot of resources is more likely to increase the makespan, as the delay will cause more unplanned resource unavailability. Therefore, they propose to weigh the free slacks of all jobs by their resource requirements. Additionally, they define a robustness measure that combines the two weights, to account for the number of successors *and* the resource requirements.

Chtourou and Haouari argue that if a job has a small free slack, it will already be sufficient to absorb a small increase in processing time. They reason that summing up large free slacks results in a disproportional measure of the ability of a schedule to deal with small increases in processing times. Therefore, they define alternatives of their measures, where they replace the free slack by a binary variable indicating if the job has free slack or not. This way, the measures only consider is a job has free slack, and give no importance to its length. Another variant they propose is replacing the free slack by the minimum of the free slack and a fraction of the processing time. They set the fraction equal to the average percentage of increase in processing time. This way, they only consider the part of the free slack that would potentially be useful.

Chtourou and Haouari develop a method that is based on a priority-rule heuristic. The heuristic is run multiple times with the objective of minimizing the makespan. The makespan found in this stage as then used as a threshold for the second stage. The heuristic is run again multiple times to maximize the robustness as defined by the robustness measures, while keeping the makespan equal to or smaller than the threshold.

Chtourou and Haouari show with a simulation study that schedules created with their method perform better than non-robust schedules in terms of quality robustness. The following robustness measures give the best results: sum of free slacks, sum of free slacks

weighted by number of successors, sum of binary value indicating if free slack exists and sum of binary value indicating if free slack exists weighted by amount of resources needed.

2.1.2 Robustness measures based on probability distribution

Canon and Jeannot (2009) perform an experimental study of different robustness measures and they propose several heuristics to minimize the makespan while maximizing quality robustness. Canon and Jeannot define the robustness of a schedule as its ability to absorb some increases in job processing times while maintaining a stable solution. They give a method to compute the makespan distribution. With this, they aim to schedule the jobs such that the average makespan is minimized and the stability of the makespan is maximized. They investigate the correlation between several robustness measures and the stability of the makespan.

An intuitive measure is the makespan standard deviation. Canon and Jeannot reason that this measure is related to the robustness because the standard deviation indicates how likely it is that the realized makespan is close to the average makespan. Bölöni and Marinescu (2002) define the *makespan differential entropy*, which is a way to assess the uncertainty of the distribution. Canon and Jeannot argue that the uncertainty of the makespan could be a good indicator of the robustness as less uncertainty indicates a higher probability of realizing the average makespan. They also consider the slack mean, which is the expected value of the sum of the free slacks. Shestak, Smith, Siegel, and Maciejewski (2006) define the probability that the makespan is between two bounds. Canon and Jeannot propose two variants as robustness measures, where they choose the bounds as the absolute or relative differences to the average makespan. Shi, Jeannot, and Dongarra (2006) define the *lateness likelihood* as the probability that the makespan exceeds a given target. Canon and Jeannot investigate this measure with the average makespan as the target. Finally, they consider the makespan 0.99-quantile. This measure indicates what the worst makespan will be in 99% of the cases.

Canon and Jeannot perform a simulation study with these robustness measures to determine the relationship between them. They show that almost all measures are equivalent. Therefore, Canon and Jeannot argue that the simplest measure, the makespan standard deviation, will be sufficient for most real cases. The outlier is the slack mean, which they show has a low and a negative correlation with the other measures. Canon and Jeannot explain this by the observation that minimizing makespan and maximizing the slack mean are conflicting objectives.

Simulation as robustness measure

van den Akker et al. (2013) use a combination of local search and simulation to find good quality robust solutions for the job shop scheduling problem. Their local search method is based on simulated annealing. In this local search method, they compare candidate solutions based on the results of a discrete event simulation. They introduce two variants of applying simulation to approximate the expected makespan.

The first variant is called *result sampling*. They run a discrete event simulation for a number of times to find a realization of a schedule. The results of these simulation runs are then averaged. In the second variant, called *cut-off sampling*, they do not use the results of all the simulation runs. van den Akker et al. argue that it is very hard to be robust against realizations of schedules where the processing times of jobs are very far from the mean. Therefore, they ignore the realizations with the smallest and largest makespan obtained from the simulations and use the remaining schedules to compute the average.

van den Akker et al. perform several experiments to compare their methods to classical methods. These classical methods use a deterministic algorithm where the stochastic variables are replaced by a percentile of the mean or by multiplying the mean with a certain factor. They show that their methods outperform classical methods.

Approximating the makespan distribution

Passage, van den Akker, and Hoogeveen (2016) present an iterated local search approach for the deterministic parallel machine scheduling problem to minimize the makespan. To apply this algorithm to the problem with stochastic processing times, they aim to minimize the expected makespan (quality robustness). They present several methods to estimate the makespan to compare candidate solutions of their local search.

Passage et al. adapt the result sampling approach of van den Akker et al. (2013) for the parallel machine scheduling problem and their local search algorithm. However, they show that it requires many samples to get an accurate estimate. As the local search approach evaluates many candidate solutions, they need a faster method. Therefore, Passage et al. present two algorithms to approximate the distribution of the makespan. In the first method, they start by approximating the makespan without precedence relations. Then, they add the approximated delay that will be caused by the precedence relations.

In their second method, Passage et al. consider the precedence constraints directly. They use dynamic programming to estimate the start time distributions of jobs by computing the maximum over the completion time distributions of all predecessors. To do this, they use the work of Nadarajah and Kotz (2008) to compute the maximum of two normal distributions.

Passage et al. perform several computational experiments and show that their approach of approximating the makespan with dynamic programming, performs as good as estimating the makespan with 300 simulation runs, while being much faster.

2.2 Simulation studies on robustness measure quality

In this section, we discuss simulation studies performed to investigate the correlation between several robustness measures and several definitions of true robustness.

2.2.1 Robustness for the RCPSP

Khemakhem and Chtourou (2013) provide an extensive study of several existing and newly proposed robustness measures for the RCPSP. They review the measures proposed by Al-Fawzan and Haouari (2005), Kobylański and Kuchta (2007), Lambrechts, Demeulemeester, and Herroelen (2008) and Chtourou and Haouari (2008). Additionally, Khemakhem and Chtourou propose several new robustness measures. These measures are mostly weighted slack functions based on slack sufficiency. The slack sufficiency is a way to measure if the slack of a job is sufficient to absorb a delay of a percentage of its duration. They extend this idea by also including the possible delays of direct predecessors or all preceding jobs.

Khemakhem and Chtourou test the robustness measures with a five-stage approach. First, they compute a threshold makespan for a problem instance with a priority-rule-based heuristic. Then they generate a set of schedules that have a makespan equal to the makespan found in step one and compute the robustness measures of each of those schedules. They then run a simulation on the schedules with increased job duration of $\mu\%$ and compute the probability that the realized makespan of the schedule exceeds the threshold makespan. Finally, they evaluate the correlation between the robustness measures and the performance measures by comping the coefficient of determination R^2.

Khemakhem and Chtourou show that their newly proposed robustness measures outperform the others. The robustness measure with highest correlation is the slack sufficiency that takes into account the possible delay of the job itself and all its predecessors.

Robustness of shunting plans

van den Broek, Hoogeveen, and van den Akker (2018) investigate robustness measures to estimate the robustness of schedules for shunting yards. Scheduling problems for shunting

yards can be modelled as a RCPSP with deadlines. Shunting schedules are subject to uncertainties such as a delayed train arrivals or service activities taking longer than expected.

They compare several existing robustness measures, based on the work of Jorge Leon et al. (1994), Al-Fawzan and Haouari (2005), Kobylański and Kuchta (2007), Khemakhem and Chtourou (2013), Wilson, Klos, Witteveen, and Huisman (2014) and Passage et al. (2016). Additionally, they propose new robustness measures based on the slack of a path of jobs in the schedule. van den Broek et al. define the slack of a path as the maximal amount of time the jobs on the path can be delayed without exceeding the deadline of the last job on the path. They argue that it is more likely that a path will have a disruption, the more jobs there are on that path. Therefore, they define a robustness measure as the minimum of the path slacks divided by the number of jobs on the path. Additionally, they estimate the duration of a path with the assumption that the duration of each job is normally distributed. With this, they define a robustness measure as the minimum probability that a path can be completed within the deadline, over all paths.

To investigate the quality of the robustness measures, van den Broek et al. generate a large number of schedules for two real-world instances of shunting yards problems. They perform a simulation to approximate the robustness of the schedules with two performance metrics: the fraction of delayed schedules, and the average lateness of the schedules. They model the disturbances in the arrival time of a trains with a uniform distribution with the scheduled arrival time as the mean, and an interval size of 10 minutes. The service activities are modelled as a log-normal distribution with the nominal activity duration as the mean, and a standard deviation of 10% of the nominal duration.

van den Broek et al. show that the robustness measures that are based on normal approximations have a strong correlation with the robustness performance metrics. Additionally, they show that the minimum total slack, which is equivalent to the deterministic makespan of a schedule, is also highly correlated with the performance metrics. Furthermore, their results show that the free slack is poorly correlated with the robustness of shunting plans, contrary to previous results for schedules without deadlines.

2.2.2 Robustness for parallel machine scheduling

Hessey, van den Akker, and Hoogeveen (2019) study the stochastic parallel machine scheduling problem with precedence constraints. They perform a computational study to determine the quality and relations of several robustness measures. They investigate several slack-based robustness measures that combine certain slack properties by summing them, taking the average or taking the minimum. Furthermore, they investigate the normal approximation method from Passage et al. (2016).

Hessey et al. use simulation to measure several quantitive definitions of robustness. To asses the quality robustness, they estimate the following from the simulation: the average makespan, the 95-th percentile of the makespan and the coefficient of variation of the makespan. To assess solution robustness, they use the following performance metrics for the simulation: the total deviation from the planned start times of jobs and the percentage of jobs that start on their planned start time.

Hessey et al. consider two distributions for the job processing times: a normal distribution and an exponential distribution, with means p_j. Using a multi-start hill-climbing local search procedure with the objective of minimizing the deterministic makespan, they generate schedules to perform the experiments with. To determine the correlation between the robustness measures and the performance metrics from the simulation, they compute the Spearman's rank correlation coefficient.

They show that the deterministic makespan and the normal approximated makespan have a strong correlation with the expected makespan and the 95-th percentile of the makespan. Additionally, they show that there is no strong correlation between the total start delays of jobs and any of the robustness measures they investigated. Similarly, there also seems to be no strong correlation between the percentage of on time jobs and any of

the robustness measures. Hessey et al. argue that it is hard to estimate the total start delay and the the percentage of on time jobs, as they vary a lot in each simulation run. Following these results, they conclude that solution robustness is hard to estimate, even when using simulation.

In their model, the slack based measures are not good estimators for any of their definitions of robustness. Hessey et al. argue that this can be due to the fact that their testcases have no deadline and are all earliest start schedules. This causes the slack in the schedules only to be due to release dates and precedence constraints. Inserting additional slack is not considered. Slack based measures can be more effective if a deadline is considered and the goal is to distribute the jobs, given a certain machine assignment. From this, they conclude that slack based measures are not suitable to be optimized on by themselves without bounds on the makespan.

2.3 Algorithms for stable baseline schedules

Herroelen and Leus (2004) present a mathematical programming model that aims at constructing stable baseline schedules for the project scheduling problem with a due date. Their objective is to minimize the expected weighted deviation of the start times of the activities from the planned start times. In their model, they assume that exactly one disruption in an activity duration will take place. This follows the approach of Jorge Leon et al. (1994). Additionally, they assume that activities will not start before their planned start time.

Herroelen and Leus compare their model to three additional heuristics adapted to their problem by simulating the resulting schedules and computing the weighted deviation from the baseline schedule. The experiments show that their proposed mathematical programming model results in less deviation than the three other heuristics, for any number of disruptions in the activity durations.

Lambrechts et al. (2008) present a tabu search approach for the RCPSP with the aim of maximizing the schedule's stability considering uncertainty in the resource availabilities. The objective function in the tabu search procedure is a surrogate robustness measure that consists of an instability weight for each job and a free slack utility function that gives decreasing returns per extra unit of free slack allocated to a job. This objective function is penalized by the the amount that the deadline is exceeded, weighted by the number of iterations that no improvement was found. The procedure uses two neighbourhoods. The first neighbourhood operator is the of swapping two jobs. The second neighbourhood operator consists of increasing a job's buffer with a discrete value between $-\Delta$ and Δ. Δ varies during the operation of the algorithm, starting with a lower value and increasing it when no improvements can be found. For each neighbourhoods they use a different iteration type, which are applied in an alternating way. Lambrechts et al. show with a simulation experiment the their procedure performs better than traditional approaches that do not allow insertion of buffers, and approaches that use simpler buffering heuristics.

Van de Vonder, Demeulemeester, and Herroelen (2008) develop several heuristic procedures to generate stable baseline schedules for the RCPSP subject to a project due date. They adopt a two-stage approach in which the problem is first solved without "protection", and buffers are added afterwards in the second stage. They propose three buffer allocation algorithms. Their first heuristic, virtual activity duration extension (VADE), computes modified durations of the activities based on their standard deviation. These adjusted durations are then used to construct the baseline schedules. Secondly, their starting time criticality (STC) heuristic combines information about the processing time variability and the activity weights. Finally, they propose a tabu-search procedure that starts with the schedule obtained with the STC-heuristic. They perform a simulation study and determine that the STC heuristic generally performs the best.

Liang, Cui, Hu, and Demeulemeester (2020) present a bi-objective optimization model for the RCPSP with the objectives of minimizing the makespan while maximizing the stabil-

ity of a schedule subject to processing time variability. The objective function consists of the normalized combination of the STC and the makespan. They propose two-stage algorithm in which they generate robust resource allocations in the first stage, and use a simulated annealing algorithm to optimally insert buffers in the second stage. In this simulated annealing algorithm, the neighbourhood operator increases the buffer of a job with a value between $-\Delta$ and Δ, following the work of Lambrechts et al. (2008). Liang et al. perform several experiments and show that their proposed two-stage algorithm is effective to create schedules that are both quality and solution robust.

2.4 Conclusion

In this chapter we looked into literature on several robustness measures. There does not seem to be consensus on the definition of true robustness and on which robustness measures are good estimators of the true robustness. Some of the measures are purely slack-based, meaning that they don't require information about the job processing time distribution. These measures base their approximation on the total slack or free slack of a schedule with a certain slack function such as taking the sum, weighted sum, or minimum. Other measures use information about the uncertainty to approximate the true robustness. Even though these measures may provide a more accurate estimate of the robustness, their assumption that information about the probability distribution of the processing times is know may limit their use in real life applications.

Furthermore, we discussed several studies performed to examine the quality of robustness measures. The results of these studies were inconsistent, demonstrating the difference in performance of robustness measures for different scheduling problems and definitions of robustness. Therefore, we will investigate the quality of the robustness measures for buffered schedules for the stochastic parallel machine scheduling problem with a global deadline constraint, with the aim of creating solution robust schedules that respect their deadline.

Chapter 3

Robustness Measures

In this chapter we define the robustness measures that we will evaluate. These measures are based on the literature reviewed in Chapter 2. Additionally, we propose new robustness measures.

3.1 Slack based measures

Sum of slacks

Jorge Leon et al. (1994) define the average total slack as a robustness measure. Since we are interested in comparing the robustness of schedules of the same instance, the number of jobs is a constant. Therefore, we compute the sum of total slacks for schedule S.

$$RM_1(S) = \sum_j TS_j^S \tag{3.1}$$

Similarly, Al-Fawzan and Haouari (2005) define the sum of free slacks as a robustness measure.

$$RM_2(S) = \sum_j FS_j^S \tag{3.2}$$

Minimum slack

Kobylański and Kuchta (2007) argue that maximizing the *minimum* slack will ensure that the slack of all jobs will be maximized, which means all jobs are protected against delays.

We compute the minimum total slack. This measure is equivalent to the deterministic makespan, as the minimum total slack is the total slack of the critical path. This is equal to the difference between the deterministic makespan and the deadline.

$$RM_3(S) = \min_j\{TS_j^S\} \tag{3.3}$$

Kobylański and Kuchta (2007) define the minimum of the free slack/processing time ratios as a robustness measure.

$$RM_4(S) = \min_j\{FS_j^S/p_j\} \tag{3.4}$$

Bounded free slack

Chtourou and Haouari (2008) argue that summing all free slacks will wrongfully inflate the usefulness of additional slack. Therefore, they propose to use the sum of the minimum over

the free slacks and a fraction of the processing time.

$$RM_5(S) = \sum_j \min\{FS_j^S, \lambda p_j\} \tag{3.5}$$

where $0 < \lambda < 1$. Chtourou and Haouari suggest to set λ to the expected percentage increase in the processing time of the job.

Alternatively, they propose summing a binary value indicating if free slack exists.

$$RM_6(S) = \sum_j \alpha_j \qquad \text{where } \begin{cases} \alpha_j = 1 & \text{if } FS_j^S > 0 \\ \alpha_j = 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Weighted free slack

Chtourou and Haouari (2008) define several robustness measures by weighing the free slack of a job in various ways. They look at the RCPSP and weigh the free slack of a job with its resource requirements. Since we do not consider resource requirements, we use the processing time of the jobs as the weight instead. A job that has a long processing time is more likely to face a (longer) delay. When such jobs have a larger free slack, it is more likely that the delay can be absorbed. Therefore, we sum the free slacks weighted by the processing time of a job.

$$RM_7(S) = \sum_j FS_j^S \times p_j \tag{3.7}$$

A job that has a large number of predecessors is more likely to experience a delayed start time caused by a delayed preceding job. When such jobs have a larger free slack, it is more likely that the start delay can be absorbed and will not propagate further. Therefore, we define a measure that sums the free slacks weighted by the number of direct predecessors.

$$RM_8(S) = \sum_j FS_j^S \times NDP_j^S \tag{3.8}$$

where NDP_j^S is the number of direct predecessors of job j in schedule S.

Chtourou and Haouari (2008) define a measure as the sum of the free slacks weighted by the number of direct successors. When a job that has a large number of successors is delayed, it will influence the start time of more succeeding jobs. When such jobs have a larger free slack, it is more likely that the delay can be absorbed and succeeding jobs can still start on time.

$$RM_9(S) = \sum_j FS_j^S \times NDS_j^S \tag{3.9}$$

where NDS_j^S is the number of direct successors of job j in schedule S.

Chtourou and Haouari (2008) combine weighing the free slack with the resource requirements and the number of direct successors. Therefore, we propose a measure where we combine RM_7 and RM_9 and weigh the free slack by the processing time and the number of direct successors.

$$RM_{10}(S) = \sum_j FS_j^S \times p_j \times NDS_j^S \tag{3.10}$$

Slack sufficiency

One variant of the slack sufficiency measure proposed by Khemakhem and Chtourou (2013) is defined as the capacity of jobs to absorb the possible delay in the job itself or in one of

the jobs that precede it. For each job j, we count how many times the free slack of job j is at least a fraction λ of the processing time of the job itself or one of the jobs that precede it:

$$RM_{11}(S) = \sum_j |\{i \mid i \in prec_j^S \cup \{j\}, FS_j^S \geq \lambda p_i\}| \tag{3.11}$$

where $prec_j^S$ are all jobs preceding job j in schedule S, i.e job i must be finished before job j can start. This includes the machine predecessors. Khemakhem and Chtourou suggest that λ should be set to the expected percentage increase in the processing time of the job.

If a job i has no given precedence constraint to job j, and i is assigned before j on the same machine, an additional predecessor $i \in prec_j^S$ is introduced. Counter-intuitively, schedules with more precedence relations may get higher robustness values from RM_{11}. This can stimulate placing jobs with precedence constraints on different machines, so that additional precedence relations are introduced. To combat this, we define the complement of RM_{11}, counting how many times the free slack is *not* sufficient. As this will give a higher value for less robust schedules, this can be seen as the costs for being non-robust.

$$RM_{12}(S) = \sum_j |\{i \mid i \in prec_j^S \cup \{j\}, FS_j^S < \lambda p_i\}| \tag{3.12}$$

Interval schedule

Wilson et al. (2014) propose a measure to determine the flexibility of a Simple Temporal Network. In this measure we want to assign an interval to each job. The goal is to find the assignment of intervals to activities in such a way that each job can be started within its interval, and the total length of the intervals is maximal. We adapt it to use the planned start time PST instead of the earliest start time EST. The intervals can be computed with this linear program.

$$\begin{aligned} RM_{13}(S) = \max \quad & \sum_j (l_j - e_j) \\ \text{s.t.} \quad & PST_j^S \leq e_j \leq l_j \leq LST_j^S && \forall j \\ & l_j + p_j \leq e_i && \forall j \prec i \end{aligned} \tag{3.13}$$

van den Broek et al. (2018) change the objective of the linear program to maximize the minimum interval. This will make the interval schedule more evenly distributed. Again, we adapt it to use the planned start time PST instead of the earliest start time EST.

$$\begin{aligned} RM_{14}(S) = \max \quad & \min_j (l_j - e_j) \\ \text{s.t.} \quad & PST_j^S \leq e_j \leq l_j \leq LST_j^S && \forall j \\ & l_j + p_j \leq e_i && \forall j \prec i \end{aligned} \tag{3.14}$$

3.2 Normal approximation

Passage et al. (2016) present a method to approximate the schedule makespan distribution based on the work of Nadarajah and Kotz (2008) for calculating the maximum of two normal distributions. To do this efficiently, we can use dynamic programming by evaluating the jobs in topological order, and computing the distribution of the start time ST_j and the completion time CT_j of job j. We compute ST_j as the maximum over the planned start time PST_j and the maximum of the completion time distribution over all direct predecessors.

$$ST_j = \max\{PST_j, \max_{i \prec j} CT_i\} \tag{3.15}$$

From this, we can compute the completion time distribution CT_j of job j. This computed by adding the processing time distribution D_j to the start time distribution ST_j.

$$CT_j = ST_j + D_j \tag{3.16}$$

van den Broek et al. (2018) then define a robustness measure as the probability that the last job is finished before the deadline.

$$RM_{15}(S) = P(\max_j CT_j \leq d) \tag{3.17}$$

Following this work, we define a new robustness measure based on the start time distributions ST_j of the jobs. We sum over each job the probability that the job will be started on time, considering its planned starting time.

$$RM_{16}(S) = \sum_j P(ST_j \leq PST_j^S) \tag{3.18}$$

3.3 Measures based on predecessor slack

The free slack of a job has no influence on its own probability to start on time. A job may be delayed if its predecessors take longer then expected. To measure solution robustness, we therefore need to consider the slack of the predecessors of a job.

We define a robustness measure that computes for each job the fraction of its predecessors that have enough free slack to absorb their expected delays.

$$RM_{17}(S) = \sum_j \alpha_j \text{ where } \begin{cases} \alpha_j = \dfrac{|\{i \mid FS_i^S \geq \lambda p_i, i \in dprec_j^S\}|}{NDP_j^S} & \text{if } NDP_j^S > 0 \\ \alpha_j = 1 & \text{otherwise} \end{cases} \tag{3.19}$$

where $dprec_j^S$ are all direct predecessors of job j in schedule S and λ is the expected percentage increase in the processing time.

Alternatively, we can estimate how much start delay we expect for each job based on the free slack of its direct predecessors. For each job j we compute its expected start delay ESD_j by evaluating the jobs in topological order.

$$ESD_j^S = \max_{i \prec j}\{\max\{\lambda p_i + ESD_i^S - FS_i^S, 0\}\} \tag{3.20}$$

where $i \prec j$ indicates that i is a direct predecessor of j. The robustness measure is then defined as the sum of all estimated start delays.

$$RM_{18}(S) = \sum_j ESD_j^S \tag{3.21}$$

Chapter 4

Robustness Measure Evaluation

In this chapter we perform experiments to determine the quality of the robustness measures presented in Chapter 3. The goal of these experiments is to determine how well each robustness measure is able to approximate the quality and solution robustness of schedules with buffers and a deadline. We assess the quality of the estimations by computing the correlation between the robustness measures and the results of a Monte Carlo simulation on a large set of instances and various performance metrics.

4.1 Experimental Setup

4.1.1 Problem instances

The difficulty in evaluating robustness measures is that they are only suitable for comparing schedules for the same problem instance. However, the experiment must be representative for the problem in general, not just for one problem instance. Therefore, we repeat the experiments for various instances.

We use the same problem instances as Passage et al. (2016). These are 12 instances, titled nJ-rR-mM where n is the number of jobs, r is the number of precedence constraints and m is the number of machines. Additionally, we generate a second set of 12 problem instances, titled nJ-rR-mM-2, for the same combinations of n, r and m as the first set of instances. This is to test the robustness measures more extensively, as instances with the same parameters can still give different results. We generate these problem instances in the same way as Passage et al.: Processing times of the jobs are randomly chosen in the interval $[1, 20]$. Release dates are chosen randomly from the interval $\lfloor 0, n/2 \rfloor$. We select p precedence relations at random but such that no cycle occurs.

4.1.2 Setting the Deadline

The instances of as Passage et al. (2016) do not have a deadline. Since we consider the problem with a global deadline constraint, we have to set a certain deadline for each instance. This deadline needs to be at least the minimal makespan for the instance, otherwise it would never be possible to find a feasible schedule. Therefore, we need a lower bound on the makespan.

Summing processing times A simple lower bound on the makespan is to take the sum of processing times divided by the number of machines. To account for the release dates, we add the m smallest release dates. This gives us:

$$minimalLength = \frac{\sum_j \{p_j\} + m \text{ smallest } r_j}{m}$$

Van de Vonder, Demeulemeester, Herroelen*, and Leus (2006) showed that a due date setting of $1.3 \times C_{max}$ is suitable for a stable schedule. Therefore, we scale *minimalLength* with 1.3 to account for the uncertain processing times.

Critical path Another lower bound on the makespan is the critical path length *cpLength*. A common way to account for the uncertainty is by increasing the length with 50% (Van de Vonder et al., 2006). However, the standard deviation of the critical path length depends on the number of jobs on the path *cpJobs*. The more jobs on the critical path, the lower the standard deviation becomes. Therefore, we scale this 50% by dividing by $\sqrt{cpJobs}$.

Deadline setting If the critical path is long, the makespan is more likely to be determined by *cpLength*. If it is short, the makespan is more likely to be determined by the *minimalLength*. Therefore, we set the deadline d to the maximum of the two:

$$d = \max\left\{cpLength \times \left(1 + \frac{0.5}{\sqrt{cpJobs}}\right), minimalLength \times 1.3\right\}$$

4.1.3 Schedule Generation

Since robustness measures will be used to evaluate the robustness of a variety of schedules, robust and non-robust, we also need to test them on a diverse set of schedules for a given problem instance. Therefore, we follow the approach of Hessey et al. (2019) of using a local search procedure with random elements to ensure that we get diverse assignments of jobs to machines. This is done by generating schedules with the objective of minimizing the makespan, assuming that all processing times are deterministic. We use a simple multi-start local search algorithm in which we run a hill-climbing local search multiple times to obtain s unique schedules (Algorithm 1). The local search is intentionally not very advanced, since the goal is to find a diversity of schedules, not the best possible schedule. However, we do not accept solutions that have a makespan larger than the deadline.

Algorithm 1 For a given schedule problem instance P, find s schedules

```
1: procedure MultiStartLS(P, s)
2:     schedules ← ∅
3:     for i ← 0 to s do
4:         repeat
5:             S ← GetInitialSolution(P)
6:             S ← HillClimbing(S)
7:         until C_max(S) ≤ d and schedules does not contain S
8:         Add S to schedules
9:     return Schedules
```

The hill-climbing procedure (Algorithm 2) is based on the work of Hessey et al. (2019). In this procedure, we consider neighbour solutions that are obtained by swapping jobs with its machine predecessor (N_0), and by moving jobs to any feasible position (N_1). When exploring a neighbourhood, the first improving neighbour is accepted as the new schedule. Therefore, the neighbours from a neighbourhood are explored in a random order. Neighbourhoods are explored in an alternating way, starting with N_0. When no improvement can be found with any of the neighbourhood operators, the schedule is returned.

Algorithm 2 For a given initial solution S, iteratively improve the solution by swapping or moving jobs

```
1: procedure HillClimbing(S)
2:     k ← 0
3:     repeat
4:         S' ← first improving neighbour from exploring N_k in random order
5:         if S' exists then
6:             S ← S'
7:         else
8:             k ← (k + 1) mod 2                    ▷ go to next neighbourhood
9:     until no improvement exists
```

An initial solution is obtained with a greedy algorithm (Algorithm 3) by repeatedly selecting the job that can start the earliest from all jobs without unassigned predecessors and scheduling it on the machine with the earliest completion time. If there are multiple candidate jobs or machines that fulfil these criteria, we choose a random job or machine from the candidates.

Algorithm 3 For a given problem instance P, greedily create a feasible schedule

```
1: procedure GetInitialSolution(P)
2:     while unassigend job exists do                ▷ Random tie-breaking
3:         j ← job without unassigned predecessors that can start earliest
4:         m ← machine with earliest completion time
5:         schedule job j on machine m
```

Inserting buffers

Schedules generated by the local search approach will be earliest start schedules, as the algorithm aims to minimize the makespan. Since our final goal is to assign specific start times to jobs by inserting buffers into the schedule, we also need testcases with varying buffers between jobs.

When inserting buffers we need to consider the deadline, because the testcases will not be interesting in terms of quality robustness if they all cannot finish within the deadline. Therefore, we use the LP from RM_{14} (equation 3.14) to compute maximum intervals for jobs. As this LP maximizes the minimum interval, the intervals for jobs will be evenly distributed. This will give us the maximum amount of buffer we can assign to each job.

To insert buffers in a diverse way for a given schedule, we randomly choose buffers between a lower bound and an upper bound and repeat this r times. These lower and upper bounds are percentages of the computed maximum buffers. We start with a lower bound of 0 and a upper bound of 0.1, and continue to increase the upper bound with 0.1 until it is 1. Then, we start increasing the lower bound with 0.1 until it is 0.9. This gives us 19 different intervals. Additionally, we include the schedule without buffers and the schedule with maximal buffers. This way, we get $19r + 2$ different buffer assignments for a given schedule.

Algorithm 4 For a given earliest start schedule S, generate different schedules by allocating varying buffers

```
1: procedure INSERT BUFFERS(S, r)
2:     schedules ← {S}                                  ▷ Include schedule without buffers
3:     maxBuffers ← GETMAXINTERVALS(S)                  ▷ from LP (equation 3.14)
4:     intervals ← {(0, 0.1), (0, 0.2), ..., (0, 1)} ∪ {(0.1, 1), (0.2, 1), ..., (0.9, 1)}
5:     for all (min, max) ∈ intervals do
6:         repeat r times
7:             for all jobs j do
8:                 buffers_j ← random(min × maxBuffers_j, max × maxBuffers_j)
9:             S' ← INSERTBUFFERS(S, buffers)
10:            add S' to schedules
11:    S' ← INSERTBUFFERS(S, maxBuffers)    ▷ include schedule with maximal buffers
12:    add S' to schedules
13:    return schedules
```

4.1.4 Simulation

We perform a Monte Carlo simulation to accurately estimate the true robustness. A schedule is simulated by visiting the jobs in topological order and computing the realized start times based on the processing times drawn from the given probability distribution D_j.

To determine the robustness of a schedule based on the simulation, we record the following performance metrics:

- The average realized makespan (quality robustness).
- The fraction of samples that was completed within the deadline (quality robustness).
- The average fraction of jobs that started on time (solution robustness).
- The average sum of job delays (solution robustness).

4.1.5 Processing time distributions

We use the following distributions D_j for the job processing times:

- Normal distribution with mean $= p_j$ and standard deviation $= \alpha \times p_j$ (N_{25} denotes $\alpha = 0.25$)
- Log-normal distribution with mean $= p_j$ and standard deviation $= \alpha \times p_j$ (LN_{25} denotes $\alpha = 0.25$)
- Exponential distribution with mean $= p_j$ (Exp)

Some of the robustness measures use a parameter λ. Chtourou and Haouari (2008) and Khemakhem and Chtourou (2013) suggest that λ should be set to the expected percentage increase of the processing time. A way to estimate this is by taking a certain percentile of processing time distribution. In the experiments of van den Akker et al. (2013), taking the 70th percentile gave the best result. Therefore, we set the expected increase to $F^{-1}(0.7) - p_j$, where $F^{-1}(x)$ is the inverse cumulative distribution function for a given distribution D_j.

4.1.6 Realizing start times

Since the processing times are stochastic, a job may not be able to start on its planned start time due to delayed predecessors. In that case, we use a policy where the job will be delayed as much as necessary while remaining on the same machine and in the specified order.

On the other hand, a job may be able to start earlier than planned because it predecessors were finished earlier than expected. In this case, we can either allow the job to start earlier than its planned start time (shift to the left), or let the job wait until its planned start time. Starting earlier is only useful for earliest start schedules with the goal of minimizing the makespan. Starting earlier in schedules with buffers will defeat the purpose of assigning specific start times. The buffers are inserted to improve the stability of the schedule execution. Letting a job start earlier will make the schedule deviate more from the original schedule, decreasing solution robustness. Therefore, we will not allow jobs to start earlier, and treat the planned start times equivalent to release dates for the computation of the robustness measures and for the simulation.

4.1.7 Steps of experiments

To determine the quality of the robustness measures, we execute the following 4 steps for each problem instance:

1. We generate 10 earliest start schedules using the multi-start local search approach (Algorithm 1 with $s = 10$). We then insert buffers into each ESS using the buffer insertion approach (Algorithm 4) with $r = 5$, which will result in $19 \times 5 + 2 = 97$ schedules per ESS (see Section 4.1.3). Therefore, we obtain $10 \times 97 = 970$ schedules in total for each problem instance.

2. Compute each robustness measure for the generated schedules.

3. Perform 1000 simulation runs and compute performance metrics for each processing time distribution.

4. Compute Spearman's rank correlation coefficients (Spearman's ρ) between the robustness measures and the performance metrics.

4.2 Initial evaluation of all RMs

To narrow down the selection of robustness measures and rule out bad performing measures for the rest of the experiments, we perform an initial investigation. In this first phase, we test all robustness measures with limited instances and probability distributions. We perform the experiment on the 12 instances by Passage et al. (2016) (excluding our own generated instances). These consist of:

- 6 small instances with $n = 30$, $r \in \{15, 30, 75\}$ and $m \in \{4, 8\}$
- 6 large instances with $n = 100$, $r \in \{50, 100, 250\}$ and $m \in \{6, 12\}$

The following probability distributions for D_j are used: N_{25}, LN_{25}, Exp. Next to the defined robustness measures, we consider the deterministic makespan C_{max} as a measure.

4.2.1 Results

In this section we discuss our initial findings of the correlation between the robustness measures and quality and solution robustness. A Spearman's ρ value close to -1 or 1 indicates a good correlation between the RM and the performance metric, whereas a value close to 0 indicates a poor correlation. The figures in this section show for each RM and probability distribution, a boxplot of the *absolute* values of the Spearman's ρ. The green highlighted boxes indicate a mean value of at least 0.9. The complete result tables can be found in Appendix A.

Results Quality Robustness

Figures 4.1 and 4.2 show the results for the average makespan and the fraction within deadline, respectively. It appears that the correlations when using N_{25} or LN_{25} are nearly identical. When using Exp, the performance of most robustness measures becomes worse. Furthermore, the results for estimating the average makespan and the fraction within deadline are similar.

C_{max}, RM_1, RM_3 and RM_{15} have the best overall performance on estimating both the average makespan and the fraction within deadline. C_{max} and RM_3 have identical correlations, as these are equivalent measures. RM_6, the sum of binary values indicating if free slack exists, shows a correlation of close to zero with both performance metrics and all distributions. This is because the test schedules we used are mostly schedules where we inserted buffers for all jobs, giving all jobs free slack. Only 10 out of 970 schedules per instance are schedules where no buffers are inserted. This means that RM_6 will only be able to make a distinction between the 10 schedules and the 960 others, which results in a poor correlation.

The rest of the measures show a high variability in correlation, as the range in correlations over the instances is large for both performance metrics and each distribution. RM_9 and RM_{10} show a high median correlation with both performance metrics, but have some outlier instances where the correlation is very poor. RM_2, RM_7 and RM_{13} show overall low correlations. RM_4, RM_5, RM_8, RM_{11}, RM_{12}, RM_{14}, RM_{16}, RM_{17} and RM_{18} have a better median value but a low minimum value.

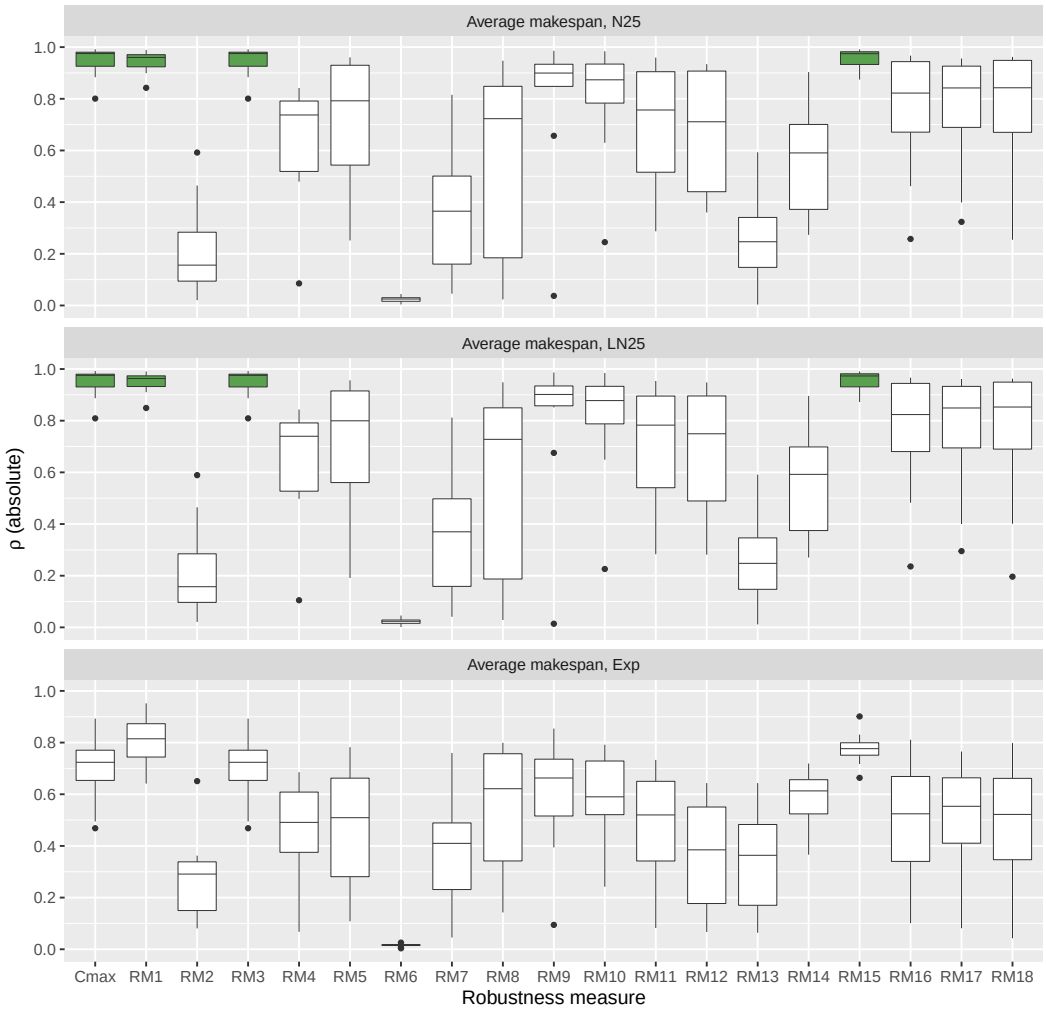

Figure 4.1: Spearman's ρ (absolute) for average makespan. Green highlight indicates mean ≥ 0.9.

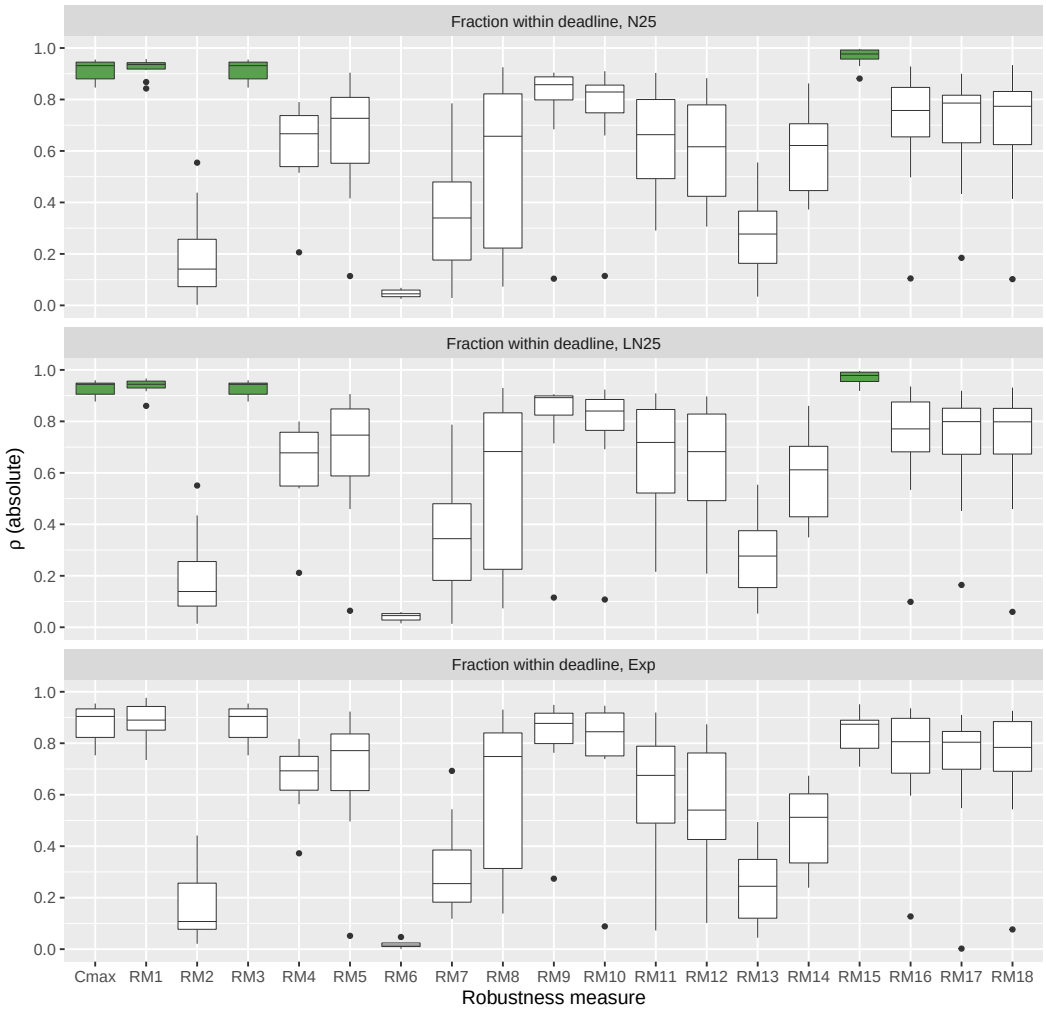

Figure 4.2: Spearman's ρ (absolute) for fraction within deadline. Green highlight indicates mean ≥ 0.9.

Results Solution Robustness

Figures 4.3 and 4.4 show the results for estimating the fraction on time jobs and the total job delay, respectively. Similar to the results for quality robustness, the correlations when using N_{25} or LN_{25} are nearly identical and the performance of most robustness measures becomes worse when using Exp. Additionally, the results for estimating the fraction on time jobs and the total job delay are very similar.

The figures indicate that RM_5, RM_9, RM_{10}, RM_{12}, RM_{16}, RM_{17} and RM_{18} have the best overall correlations (mean ≥ 0.9) with both the fraction on time jobs and the total job delay. Especially RM_{16}, RM_{17} and RM_{18} show a very high correlation for both performance metrics. RM_4 shows a relatively high overall correlation, but the maximal value does not go above 0.9. RM_{11} has a high median correlation with both performance metrics when using N_{25} and LN_{25}, but when using Exp the minimal correlation is low. Again, RM_6 shows a correlation of close to zero with both performance metrics and all distributions. The rest of the measures show a high variability in correlation for both performance metrics and each distribution. C_{max}, RM_1 and RM_3 show a high median correlation with both performance metrics, but have some outlier instances where the correlation is low. RM_2, RM_7, RM_{13} and RM_{14} show overall low correlations. RM_8 and RM_{15} have a better median value but a

low minimum value.

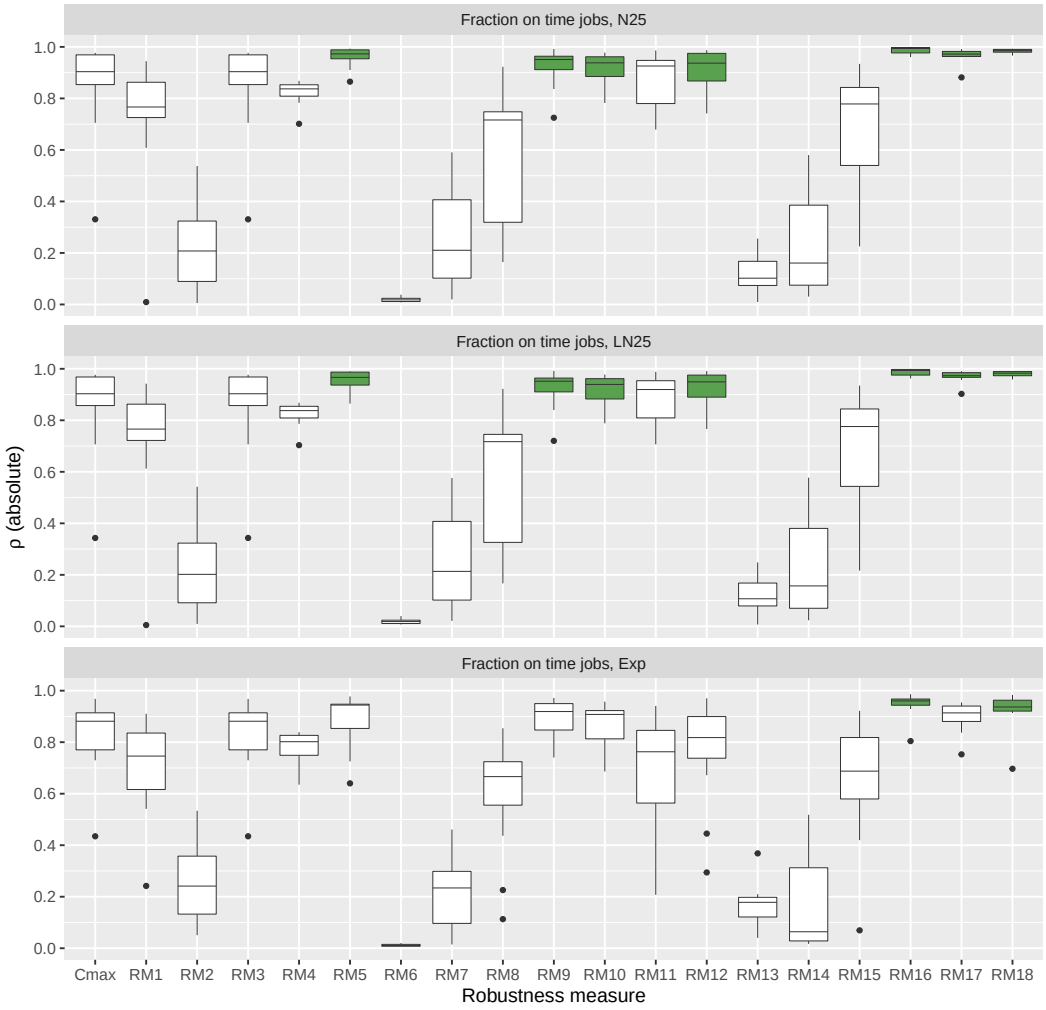

Figure 4.3: Spearman's ρ (absolute) for fraction on time jobs. Green highlight indicates mean ≥ 0.9.

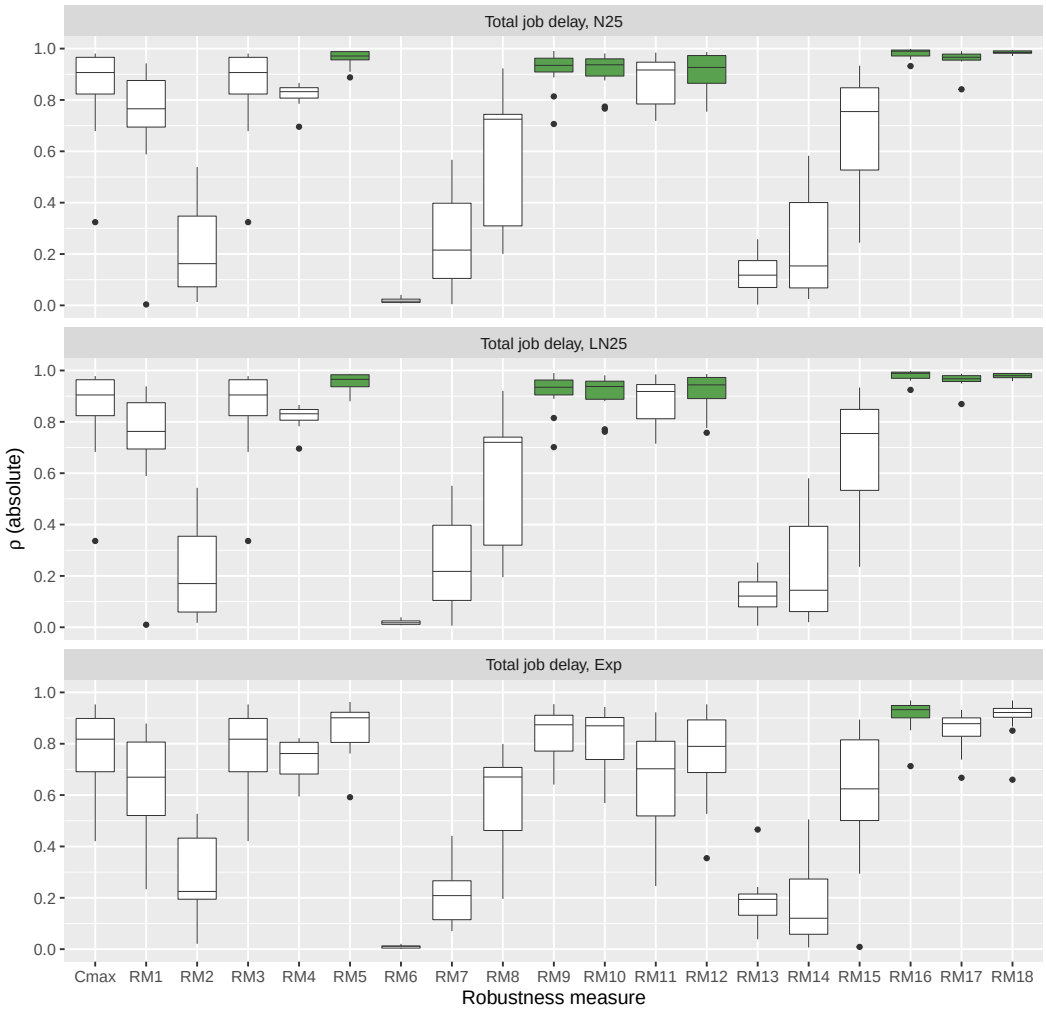

Figure 4.4: Spearman's ρ (absolute) for total job delay. Green highlight indicates mean $\geq$ 0.9.

4.2.2 Conclusion

In this section we performed initial experiments on the correlation between all robustness measures and quality and solution robustness. The high correlations of normal approximation methods RM_{15} and RM_{16} and slack-based measures RM_5, RM_{12}, RM_{17} and RM_{18}, which are based on the expected increase in processing time, showed the advantages of taking into account information about the processing time probability distributions.

Furthermore, the total slack-based measures RM_1, RM_3 are highly correlated with the quality robustness. Total slack is closely related to quality robustness, as total slack indicates how much room there is in the schedule for jobs to be delayed without exceeding the deadline. Free slack is more related to solution robustness, as free slack serves a way to absorb delays of jobs and therefore avoiding the start delay of succeeding jobs. Out of the free slack-based measures without information about the processing time variability, RM_9 and RM_{10} showed the highest correlation with solution robustness. Finally, we showed that the complement measure RM_{12} is more correlated with solution robustness than the original measure RM_{11}.

Concluding, the results showed that C_{max}, RM_1, RM_3 and RM_{15} have the highest correlation with the average makespan and the fraction on time jobs. Because C_{max} and RM_3 are equivalent measures, the correlations are identical. Therefore, we only consider

RM_1, RM_3 and RM_{15} and not C_{max} in the next experiments on quality robustness. RM_5, RM_9, RM_{10}, RM_{12}, RM_{16}, RM_{17} and RM_{18} are the highest correlating with the fraction on time jobs and the total job delay. Therefore, we will consider these robustness measures for the following experiments on solution robustness.

4.3 Evaluating Best RMs

We perform more elaborate experiments to investigate the quality of the measures that showed the best performance in the initial investigation. For these experiments we use the full set of 24 instances. These are:

- 12 small instances with $n = 30$, $r \in \{15, 30, 75\}$ and $m \in \{4, 8\}$
- 12 large instances with $n = 100$, $r \in \{50, 100, 250\}$ and $m \in \{6, 12\}$

Additionally, we introduce more variability in the processing times by also considering distributions N_{50} and LN_{50} for D_j. Finally, we record the computation time for each robustness measure to compare their computational efficiency.

4.3.1 Results Correlations

In this section we discuss the results of the correlation between the best robustness measures and quality and solution robustness. A Spearman's ρ value close to -1 or 1 indicates a good correlation between the RM and the performance metric, whereas a value close to 0 indicates a poor correlation. The figures in this section show for each RM and probability distribution, a boxplot of the *absolute* values of the Spearman's ρ. The colours of the boxes indicate the instance size.

Results quality robustness

We consider the following measures to approximate the quality robustness: RM_1 (sum of total slack), RM_3 (minimum total slack) and RM_{15} (normally approximated probability of makespan within deadline). Figures 4.5 and 4.6 show the correlations of the robustness measures with the average makespan and the fraction within deadline, respectively.

Looking at Figure 4.5, RM_{15} generally has the highest correlations with the average makespan out of the three measures. RM_1 and RM_3 have a better performance than RM_{15} for the large instances when using distributions N_{25}, LN_{25}. All three measures have a lower minimal correlation with the large instances, for almost every distribution. The correlations of RM_1 and RM_3 with the average makespan decline as we introduce more variability in processing times with distributions N_{50}, LN_{50} and Exp, especially for the large instances. The correlations of RM_{15} shows similar results for N_{25}, LN_{25}, N_{50} and LN_{50}, but are worse with distribution Exp.

Figure 4.6 suggests that RM_{15} generally has the highest correlations with the fraction within deadline out of the three measures as well. The results for the fraction within deadline shows less difference in correlations between the measures, distributions and the instance size. In general, all measures have similar correlations for N_{25}, LN_{25}, N_{50} and LN_{50}, but worse results with distribution Exp. However, RM_3 also shows some low outlier values with N_{50} and LN_{50} on the large instances. Overall, the performance of RM_1 and RM_3 is better for fraction within deadline than for the average makespan, especially for the large instances.

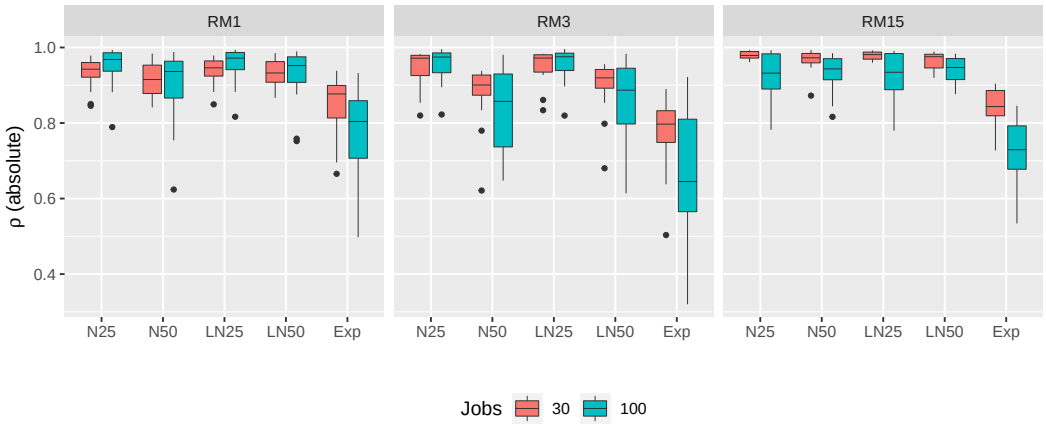

Figure 4.5: Spearman's ρ (absolute) for average makespan.

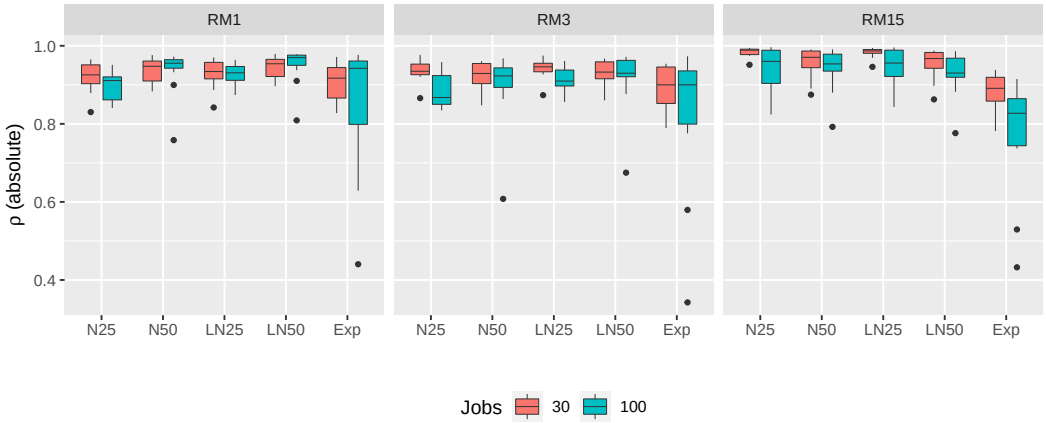

Figure 4.6: Spearman's ρ (absolute) for fraction within deadline.

Results solution robustness

To approximate the solution robustness, we consider the following measures: RM_5 (sum of minimum free slack/processing time ratio), RM_9 (sum of free slack scaled by number of successors), RM_{10} (sum of free slack scaled by processing time and number of successors), RM_{12} (slack sufficiency inverse), RM_{16} (sum of normally approximated probability of job starting on time), RM_{17} (sum of fraction of predecessors with free slack at least fraction of its duration) and RM_{18} (sum of expected delay based on free slack of predecessors). Figures 4.7 and 4.8 show the results for the fraction on time jobs and the total job delay, respectively.

The figures indicate very similar results for the two performance metrics. RM_{16} is highly correlated with both performance metrics for all distributions and instances. RM_{17} and RM_{18} also show high correlations, but have some lower outlier values. Similarly, RM_5, RM_9 and RM_{10} overall have good correlations with both performance metrics, but have outlier instances with worse values. RM_{12} has a varying performance, with high maximum values and low minimum values.

All measures show a higher median correlation value with the large instances for both performance metrics and all distributions. However, the lowest outliers of the large instances are mostly worse than the small instances. Additionally, the probability distribution is less influencing on the correlations than with the quality robustness measures.

Figure 4.7: Spearman's ρ (absolute) for fraction on time jobs.

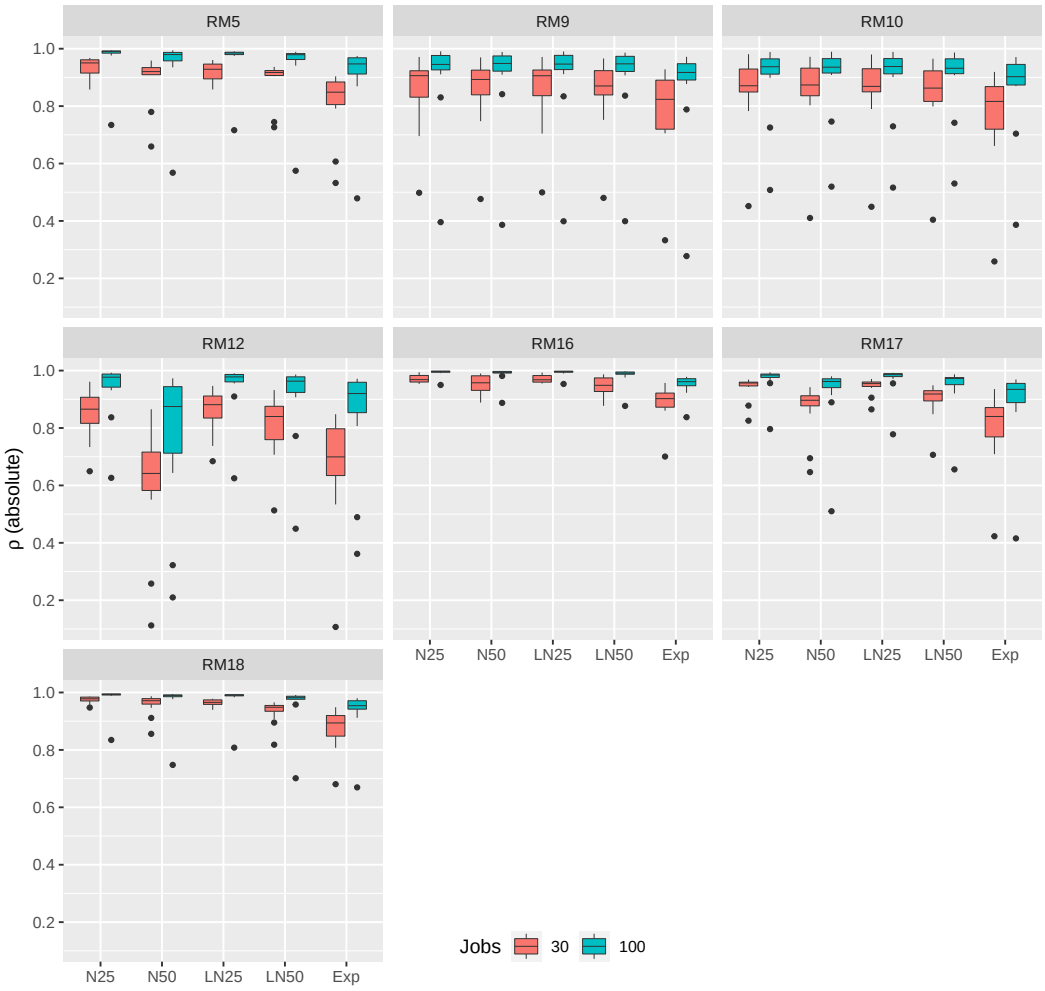

Figure 4.8: Spearman's ρ (absolute) for total job delay.

4.3.2 Computational efficiency

To compare the computational efficiency of the robustness measures, we record the total time it takes to evaluate the 970 generated schedules for the smallest and the largest instance. We repeat this 100 times and report the average total time. As a comparison, we also record the time of evaluating the 970 schedules by performing 100 simulation runs and computing C_{max}. For this experiment a laptop with an Intel® Core™ i7-4750HQ @ 2.00GHz processor was used.

The results can be seen in Table 4.1. It shows that C_{max} and slack-based measures RM_1, RM_3, RM_5, RM_9, RM_{10} are similar in terms of computation time. To compute them, the schedule has to be traversed once. RM_{17} and RM_{18} are slack based measures that take more time because the schedule has to be traversed twice to compute them. Additionally, we can see that normal approximation methods RM_{15} and RM_{16} are less computationally efficient than the simpler slack-based measures. Furthermore, the more complex slack-based measure RM_{12} has a bad efficiency in comparison to the other measures. Finally, the robustness measures all show a significantly better computational efficiency than performing 100 simulations. This is as expected, since running 100 simulations requires traversal through the graph 100 times.

	30J-15R-4M	100J-250R-12M
C_{max}	3.17	13.38
RM_1	4.30	19.40
RM_3	2.78	17.54
RM_5	4.28	18.80
RM_9	2.22	13.19
RM_{10}	2.13	12.48
RM_{12}	18.74	228.77
RM_{15}	15.17	83.46
RM_{16}	13.88	90.65
RM_{17}	4.94	39.20
RM_{18}	5.02	42.82
100Sim	362.39	1483.82

Table 4.1: Computation time in milliseconds of evaluating 970 schedules

4.3.3 Conclusion

In this section we performed more elaborate experiments to investigate the correlation of the best robustness measures with the quality and solution robustness. For these experiments we used the full set of 24 instances and introduced more variability in the processing times. Additionally, we reported the computational efficiency of the measures.

We have shown that RM_1, RM_3 and RM_{15} have high correlations with quality robustness. RM_{15} generally gives the best results. This is as expected, as this measure is formulated to approximate the probability that the makespan is within the deadline, which is exactly what the performance metric *fraction within deadline* computes. To give an accurate approximation, RM_{15} uses information on the processing time probability distributions. If this information is not known, the total slack-based measures RM_1 and RM_3 are good alternatives. RM_3, the minimum total slack, is equivalent to the makespan of a schedule for problems with a deadline. Since the makespan can be computed more efficiently than the normal approximation, it seems likely that using the deterministic makespan is sufficient to obtain quality robust schedules for problems with buffers and a deadline.

Furthermore, we have shown that the normal approximation method RM_{16} has exceptionally high correlations with solution robustness. Analogous to RM_{15}, the high performance of RM_{16} is expected since it approximates the probability that a job can start on time, which is precisely what the performance metric *fraction on time jobs* computes. Measures RM_5, RM_{17} and RM_{18} show high correlations as well, but for some instances the performance is not optimal. These four measures depend on information about the job processing time probability distributions. RM_{16} normally approximates job start times based on the processing time distribution. RM_5, RM_{17} and RM_{18} are based on the expected increase in job processing times. If we have no information about the distributions, free slack-based measures RM_9 and RM_{10} provide an alternative. These measures show overall good correlations with solution robustness, but depending on the instance, these measures could provide poor estimates.

4.4 Conclusion Robustness Measure Evaluation

In this chapter, we have studied the ability of several robustness measures to estimate the quality and solution robustness of schedules with buffers and a deadline. To this end, we compared the robustness measures with the results of a Monte Carlo simulation and reported their correlation.

We have shown that robustness measures based on the normal approximations of the job processing times are strongly correlated with robustness of the schedules, but do require more

computation time than most slack-based measures. Additionally, slack-based measures that are based on the expected increase of the processing times appeared promising as well. This shows us the advantages of having information about the variance in processing times. If this information is not know, several purely slack-based measures provide a good alternative and require less computation time, but will generally give less accurate results.

Finally, we have shown that computing the robustness measures only takes a fraction of the time of performing a sufficient number of simulation runs, illustrating the computational advantage of using robustness measures over simulation.

Chapter 5

Local search for robust scheduling

Based on the results of the simulation study, we have determined which robustness measures have the highest correlation with quality and solution robustness performance metrics. In this chapter, we present a local search algorithm in which we use a combination of these best robustness measures to determine the quality of a solution. This allows us to examine the performance of the robustness measures in a practical setting.

5.1 Algorithm Outline

In this section, we discuss our local search approach that generates buffered schedules. Our algorithm is an adaption of the local search method of Passage et al. (2016).

5.1.1 Assigning jobs to machines

Passage et al. (2016) present a local search approach for parallel machine scheduling with precedence constrains and release dates. This approach consists of an iterated local search algorithm with a variable neighbourhood descent subroutine.

Neighbourhoods

Local search is a procedure where we iteratively try to find a better solution by making (small) changes to the current solution. A new solution found by making a modification with a certain operator is called a neighbour solution. A neighbourhood is the set of all neighbours that can be found by applying a certain neighbourhood operator. Passage et al. (2016) define the following neighbourhood operators:

k-swap The swap operator selects two jobs and swaps them in the schedule. The k-swap operator selects a chain of k jobs and moves each job in the chain to the position of the next job in the chain. More formally, k jobs $j_1, j_2, \ldots, j_k$ are selected, and j_i is put at the position of j_{i+1} for $i < k$ and j_k is put at the position of j_1.

k-move The move operator selects a job and inserts it at a position different than its current position in the schedule. The k-move operator performs k successive move operations.

The neighbour solutions created by these operations may violate the precedence relations of the schedule. Therefore, only feasible neighbours are considered. A neighbour solution is regarded as feasible if all precedence relations are respected (no cycles in the schedule).

Variable Neighbourhood Descent

To combine the neighbourhood operators in one procedure, Passage et al. use a variable neighbourhood descent (VND) algorithm. In this algorithm, a sequence of neighbourhood operators $N_1, \ldots, N_{k_{max}}$ is defined, which are alternated throughout the search process. A subroutine of the algorithm repeatedly selects the first improving neighbour from the current neighbourhood N_k, while investigating at most l neighbours in a random order. If an improving solution is found, we continue with the first neighbourhood N_1. Otherwise, we continue with the next neighbourhood in the sequence N_{k+1}. The algorithm stops when all neighbourhoods are exhausted.

Iterated Local Search

To improve the local optimum found by the VND algorithm, Passage et al. apply the VND procedure as a subroutine in an iterated local search (ILS) approach. ILS is based on the principle that a better local optimum can be found by making small modifications (perturbations) to the current solution and starting the local search subroutine again with the perturbed solution as the initial solution. Passage et al. perturb the current solution by applying the k-swap operator to swap k random jobs. After the perturbation, the VND subroutine is applied again and returns a new solution. If that is better than the current solution, the ILS continues with the new solution. This is repeated for a specified number of times.

5.1.2 Inserting buffers

The local search approach described in Section 5.1.1 creates earliest start schedules: no buffers are inserted between jobs. To insert buffer into these schedules, we use a hill-climbing (HC) procedure. We adopt the neighbourhood operator from Lambrechts et al. (2008), in which the buffer of a job is increased or decreased by a discrete value between $[-\Delta, \Delta]$. The neighbourhood consists of all feasible buffer changes of all jobs. A neighbour solution is considered feasible if the resulting buffer of the job is larger than or equal to zero, and at most the available total slack of the job. In each iteration, we select the best improving neighbour as the current solution. The algorithm stops when no improvement can be found any more.

5.1.3 Objective Function

Solution robustness and quality robustness are two conflicting objectives, as inserting buffers into the schedule will increase the makespan. There are several ways to deal with this conflict. A bi-objective model can be defined which makes a trade-off between solution and quality robustness (Al-Fawzan and Haouari (2005), Liang et al. (2020)). Another approach is to set a lower bound on one of the robustness objectives, while maximizing the other objective (Chtourou and Haouari (2008), Lambrechts et al. (2008)).

We adopt the last approach, in which we aim to maximize the solution robustness, while having a lower bound on the quality robustness. However, setting the lower bound as a constraint will restrict the search space too much. Therefore, we relax the constraint and use a penalty term to represent the lower bound on quality robustness. This gives us the following objective function f to evaluate the quality of a solution:

$$f(S, pw) = O(S) - pw \times P(S) \tag{5.1}$$

in which $O(S)$ is an objective function that determines the solution robustness, and $P(S)$ is a penalty function that determines the violation of the constraint on quality robustness. pw serves as a penalty weight, which indicates the importance of the penalty value.

Adaptive penalty

Using a static penalty weight can be difficult, because it requires extensive fine-tuning to find the right value, and even then it is possible that an infeasible solution will be found. To overcome this difficulty, a dynamic penalty can be used. A dynamic penalty evolves as the search process progresses. This allows the algorithm to explore the search space at first, and as the penalty is gradually increased, the algorithm is guided towards a search space that is more likely to contain a feasible solution.

Joines, Houck, et al. (1994) propose a penalty function in the form of $pw = (\beta i)^{\alpha}$, in which i is the current iteration. To control the starting value, we introduce a fourth parameter γ, resulting in the following penalty update function:

$$\textsc{UpdatePenalty}(\alpha, \beta, \gamma, i) = (\beta \times (i + \gamma))^{\alpha}$$

5.1.4 Local Search Algorithm For Solution Robustness

We adapt the ILS algorithm of Passage et al. to include the buffer insertion HC procedure described in Section 5.1.2, and incorporate the concept of a dynamic penalty.

To combine the VND with the buffer insertion HC, we insert buffers into the resulting VND (earliest start) schedules after each execution of the VND subroutine. For each ILS iteration, all buffers are reset to zero to start with an easiest start schedule again in the perturbation step and the VND.

To apply a dynamic penalty scheme in the ILS algorithm, we follow the approach of Thomas and Manni (2014). In this approach, the penalty weight is updated before every perturbation step. This updated penalty weight is then used in the VND and HC subroutines to evaluate the objective function.

Combining these concepts results in ILS Algorithm 5, with VND (Algorithm 6) and HC (Algorithm 7) subroutines. As an initial solution for the ILS algorithm, we use the greedy algorithm (Algorithm 3) described in Section 4.1.3, which returns an unbuffered schedule.

Algorithm 5 Iterative Local Search

Input Problem instance P, perturbation amount pa, perturbation size k, objective function f, penalty parameters α, β, γ

```
1:  procedure ILS(P, pa, k, f, α, β, γ)
2:      S ← GetInitialSolution(P)                    ▷ Algorithm 3 (Section 4.1.3)
3:      i ← 0
4:      pw ← UpdatePenalty(α, β, γ, i)
5:      S* ← VND(S, f, pw)                           ▷ Obtain schedule without buffers
6:      S* ← HcBuffers(S*, f, pw)                    ▷ Insert buffers into schedule
7:      while i < pa do
8:          i ← i + 1
9:          pw ← UpdatePenalty(α, β, γ, i)
10:         S' ← ResetBuffers(S*)                    ▷ Reset all buffers to zero
11:         S' ← SwapRandomK(S', k)                  ▷ Perturb solution
12:         S'* ← VND(S', f, pw)
13:         S'* ← HcBuffers(S'*, f, pw)
14:         if f(S'*, pw) > f(S*, pw) then
15:             S* ← S'*
16:     return S*
```

Algorithm 6 Variable Neighbourhood Descent

Input schedule S, objective function f, penalty weight pw

procedure VND(S, f, pw)
  $k \leftarrow 1$
  while $k < k_{max}$ **do**
    $S' \leftarrow$ first improving neighbour in $N_k(S)$ from exploring at most l neighbours in random order.
    if S' exists **then**
      $S \leftarrow S'$
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
  return S

Algorithm 7 Hill-climbing for buffers

Input schedule S, objective function f, penalty weight pw

procedure HcBuffers(S, f, pw)
  while improvement exists **do**
    $S' \leftarrow$ best neighbour from neighbourhood generated by increasing the buffer size of a job by a discrete value between $[-\Delta, +\Delta]$
    if $f(S', pw) > f(S, pw)$ **then**
      $S \leftarrow S'$
  return S

5.2 Experimental Setup

5.2.1 Objectives and penalties

We want to examine the performance of the robustness measures as estimators of the schedule quality in the local search procedure. Therefore, we use the highest correlating robustness measures for solution robustness as the objective function $O(S)$ and adapt the robustness measures with highest correlation with quality robustness to serve as a penalty $P(S)$.

Objectives

Since we are maximizing the objective function, we need to negate the measures that have a negative correlation with robustness. Additionally, we normalize the values of the robustness measures, such that the values are independent of the problem instance. This is done by dividing by the maximal value the measure can take in an unbuffered schedule. This results in the following objectives:

$$O_{RM_5}(S) = \frac{RM_5(S)}{\sum_j \lambda p_j} \tag{5.2}$$

$$O_{RM_9}(S) = \frac{RM_9(S)}{\sum_j p_j \times \sum_j NDS_j^S} \tag{5.3}$$

$$O_{RM_{10}}(S) = \frac{RM_{10}(S)}{\sum_j p_j \times \sum_j NDS_j^S \times p_j} \tag{5.4}$$

$$O_{RM_{16}}(S) = \frac{RM_{16}(S)}{n} \tag{5.5}$$

$$O_{RM_{17}}(S) = \frac{RM_{17}(S)}{n} \tag{5.6}$$

$$O_{RM_{18}}(S) = -\frac{RM_{18}(S)}{\sum_{i=0}^{n-1} \lambda p_{J_i} \times i} \tag{5.7}$$

where J is the set of jobs in ascending order of duration.

Penalties

As penalties we consider the deterministic makespan and the normal approximation measure RM_{15}. To use the makespan to set a lower bound on quality robustness, we define a penalty that penalizes the solution when the deterministic makespan exceeds the deadline d. Since the realized makespan will most likely be larger than the deterministic makespan, having a deterministic makespan that is exactly (or very close to) the deadline will not result in schedules that can be finished before the deadline. Therefore, we define another variant where we set a bound on the makespan of 0.9 times the deadline. This factor can be decreased further to ensure a more quality robust schedule, but we perform the experiments with these two variants to demonstrate their difference.

RM_{15} approximates the probability that the schedule will finish within the deadline. To use RM_{15} to set a lower bound on quality robustness, we can define a minimal probability for the schedule to finish within the deadline according to RM_{15}. However, we found that this approach does not always guide the algorithm towards a feasible solution. This can be illustrated as follows. Usually, the makespan of the initial schedule in the local search exceeds the deadline by a significant amount such that the approximated probability of finishing within the deadline is zero. The defined neighbourhood operators can decrease the makespan by some amount, but since the deadline is exceeded by a large amount, the decrease in makespan will not be enough to improve the approximated probability of finishing within the deadline. Therefore the neighbour solution will not be accepted. This will leave the algorithm stuck at schedules with approximated probability zero of finishing with the deadline. Therefore, we reformulate the penalty of RM_{15} to take into account the makespan value itself. This is done by taking a certain percentile of the normally approximated makespan distribution, and penalizing the amount this percentile exceeds the deadline. The percentile value chosen represents the amount of safety we want to consider. For our experiments, we chose a percentile of 0.8.

We scale the penalties such that they are independent of the instance. This results in the following penalty functions:

$$P_{Cmax}(S) = \frac{max(0, C_{max} - d)}{d} \tag{5.8}$$

$$P_{Cmax0.9}(S) = \frac{max(0, C_{max} - 0.9d)}{0.9d} \tag{5.9}$$

$$P_{RM_{15}}(S) = \frac{max(0, RM'_{15}(S, 0.8) - d)}{d} \tag{5.10}$$

where $RM'_{15}(S, p)$ returns the p-th percentile of the normal approximated makespan distribution $\max_j CT_j$ of schedule S.

Baseline values

To see the influence of using the robustness measures as objectives and penalties, we also run the algorithm with the objective of deterministic makespan minimization (O_{Cmax}), without a penalty (P_{-}). Additionally, we run the algorithm where the objective function is determined by 100 simulation runs. The objective (O_{100Sim}) is the simulated fraction on time jobs, and the 80th percentile of the simulated makespan serves as a penalty (P_{100Sim})

similarly to the RM_{15} penalty. This way, we obtain an approximation of the optimal objective function, which we can use as a baseline to compare the results of the other objective functions to.

5.2.2 Parameters

The parameters for the VND and ILS are set in accordance with the results of Passage et al. (2016):

- Perturbation amount $pa = 20$
- Perturbation size $k = 5$
- Number of explored neighbours $l = 1000$
- Neighbourhoods in order: { 1-move, 2-swap, 2-move }

After experimenting with several configurations for the penalty parameters, we concluded the following values to be appropriate for our problem: $\alpha = 2$, $\beta = 1$, $\gamma = 1$. We set the buffer change parameter Δ to 3, in accordance with Liang et al. (2020).

5.2.3 General Setup

For our experiments we use the same 24 instances from Chapter 4. The local search algorithm is run for all combinations of objectives and penalties defined in Section 5.2.1. To approximate the robustness of the resulting schedules, we simulate the schedules using 1000 samples and compute the following performance metrics:

- The deterministic makespan.
- The average realized makespan (quality robustness).
- The 95th percentile of the makespan (quality robustness).
- The fraction of samples that was completed within the deadline (quality robustness).
- The average fraction of jobs that started on time (solution robustness).
- The average sum of job delays (solution robustness).

The following processing time distributions are used: N_{25}, N_{50}, LN_{25}, LN_{50}, Exp. To account for the stochasticity of the local search, we repeat the experiments 10 times and report the average values.

5.3 Results

In this section we present the results of the local search experiments. Figures 5.1, 5.2, 5.3, 5.4 and 5.5 show the results for distributions N_{25}, N_{50}, LN_{25}, LN_{50} and Exp, respectively. Since the makespan and the total delay is dependent of the problem instance, we need some sort of normalization to be able to average the results over all instances. Therefore, we use the results of the 100 simulations objective function as a baseline value, and report the relative difference with the baseline. The figures show for each objective function a boxplot over all instances. Large outliers for fraction within deadline are left out, as they give a skewed image of the true performance when the absolute values are very low (almost zero). Since normalization is not required for the performance metrics *fraction within deadline* and *fraction in time jobs* as the values are between 0 and 1 for all instances, we also report the absolute values in Figure 5.6. Full result tables for instances 30J-15R-4M and 100J-250R-12M can be found in Appendix B.

5.3.1 Makespan minimization

The results show that the deterministic makespan is lowest if deterministic makespan minimization O_{Cmax} is applied. This also results in the highest fraction within deadline and the lowest average and 95th percentile makespan when distributions N_{25} and LN_{25} are used. However, deterministic makespan minimization leads to low solution robustness in general.

5.3.2 Penalties

Quality robustness

$P_{RM_{15}}$ results in the lowest 95th percentile makespan for distributions N_{50}, LN_{50} and Exp, even when the deterministic makespan is higher than O_{Cmax} and P_{100Sim}. This indicates that RM_{15} is able to account for the stochasticity better than an approximation of 100 simulations and makespan minimization when the job processing time variability is high.

Additionally, $P_{RM_{15}}$ always gives better results than P_{Cmax} in terms of the average and 95th percentile makespan and the fraction within deadline, for all distributions and objectives. $P_{RM_{15}}$ generally also results in higher quality robustness than $P_{Cmax0.9}$, especially for the distributions with high variability. Furthermore, when $P_{Cmax0.9}$ is used, the quality robustness is always better than for P_{Cmax}. This is as expected, since putting more restriction on the makespan will leave more room to absorb delays.

Solution robustness

One might expect than since the fraction within deadline is higher for $P_{RM_{15}}$, the fraction on time jobs is lower, as these can be conflicting objectives. However, $P_{RM_{15}}$ also seems to have a positive effect on the fraction on time jobs and the total job delay with all distributions. In most cases, using $P_{RM_{15}}$ gives the highest solution robustness out of all penalties. Only with the objectives $O_{RM_{16}}$ and $O_{RM_{18}}$, P_{Cmax} gives a slightly better solution robustness. However, in that case the quality robustness is significantly worse, therefore making it a poor trade-off.

5.3.3 Objectives

Solution robustness

The results show that O_{RM_9} and $O_{RM_{10}}$ do not have an improving effect on the fraction on time jobs and the total job delay with penalty P_{Cmax} or $P_{Cmax0.9}$. The solution robustness is worse than for the other objectives aiming to optimize solution robustness. Moreover, in some cases O_{RM_9} and $O_{RM_{10}}$ perform even worse than deterministic makespan minimization O_{Cmax}, which has no incentive to optimize solution robustness.

O_{RM_5}, $O_{RM_{16}}$, $O_{RM_{17}}$ and $O_{RM_{18}}$ all result in a higher fraction on time jobs and lower total job delay than makespan minimization, for every distribution and penalty. In most cases, these measures also result in a better solution robustness than the simulation objective. $O_{RM_{16}}$ and $O_{RM_{18}}$ give the highest solution robustness out of all objectives, with $O_{RM_{16}}$ being the best overall.

Quality robustness

Comparing the RM objectives, the quality robustness seems highest with $O_{RM_{18}}$ and the worst with O_{RM_9} and $O_{RM_{10}}$. The difference is clearer with penalties P_{Cmax} and $P_{Cmax0.9}$. However, the quality robustness is in general more dependant on the penalty than on the objective.

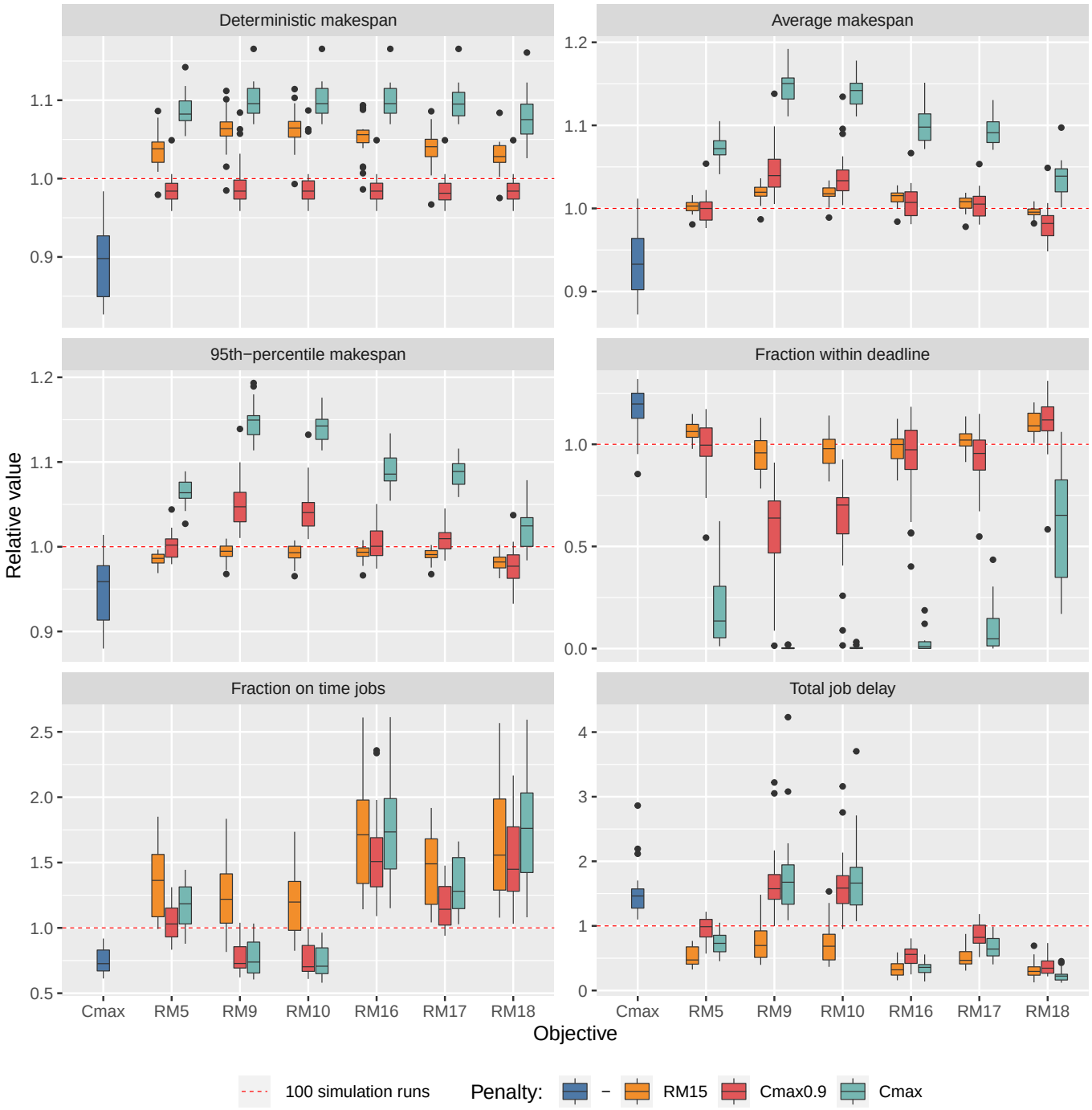

Figure 5.1: Results for distribution N_{25}. Values are relative to the results of 100 simulation runs in the objective function.

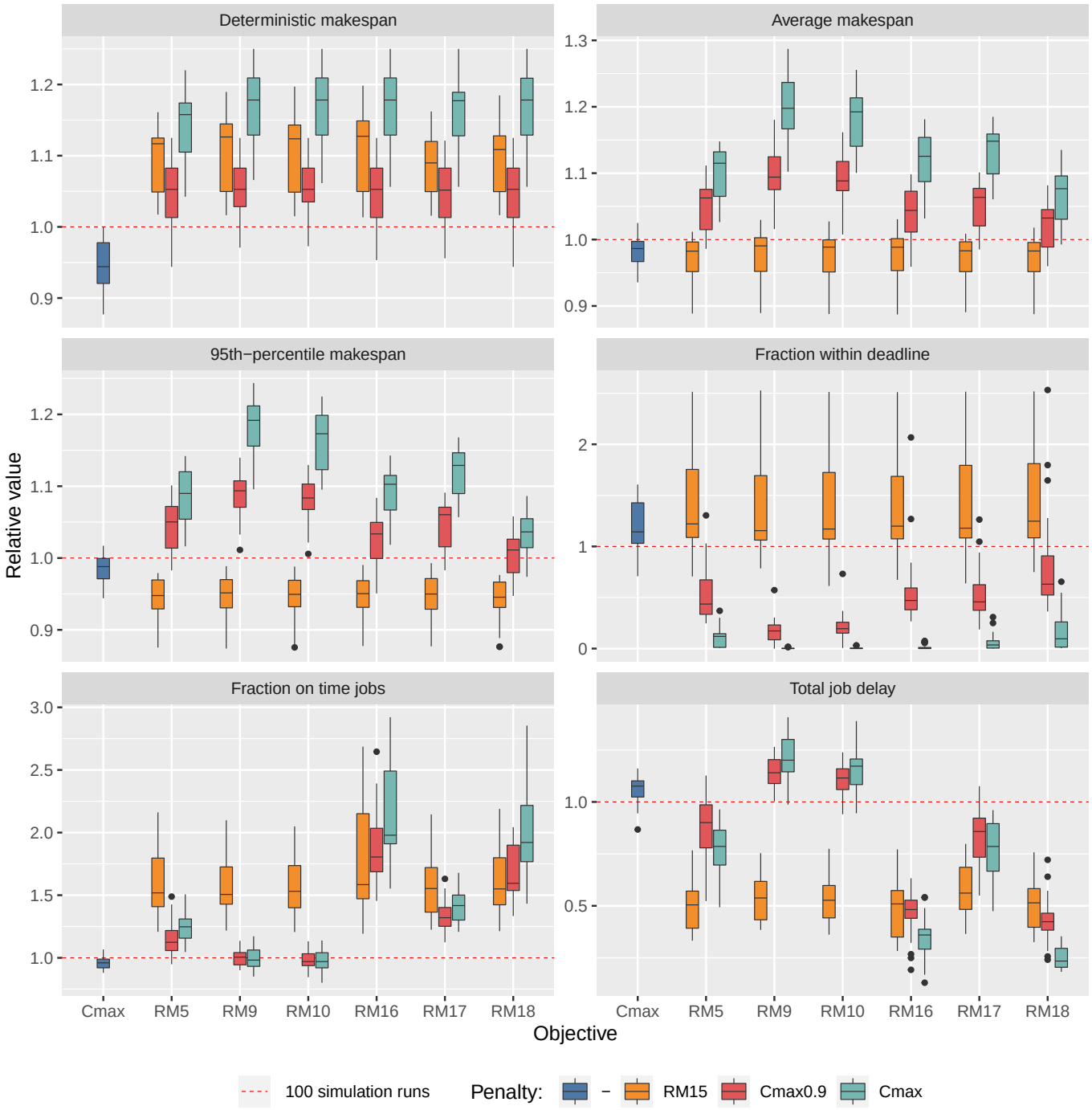

Figure 5.2: Results for distribution N_{50}. Values are relative to the results of 100 simulation runs in the objective function.

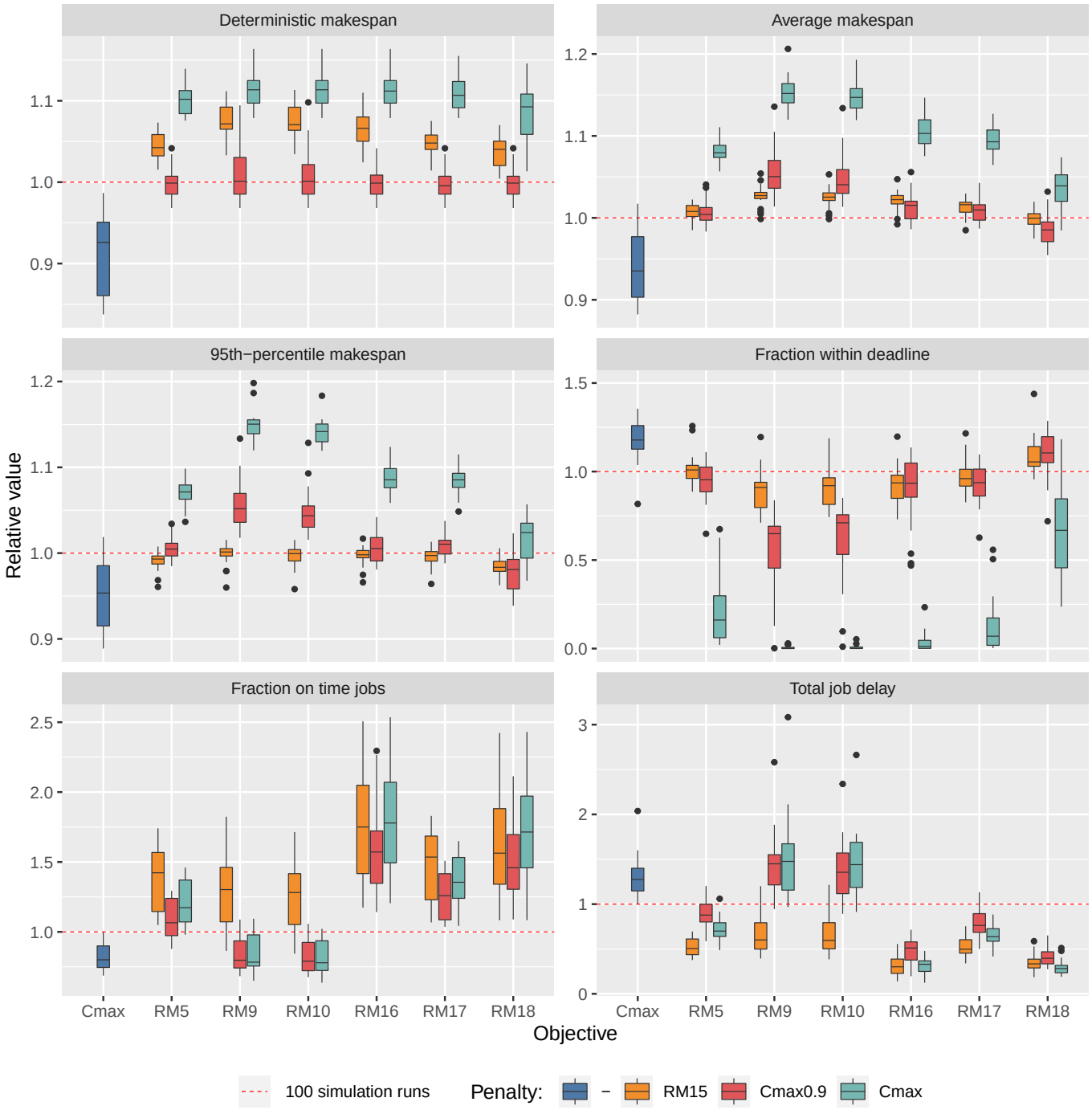

Figure 5.3: Results for distribution LN_{25}. Values are relative to the results of 100 simulation runs in the objective function.

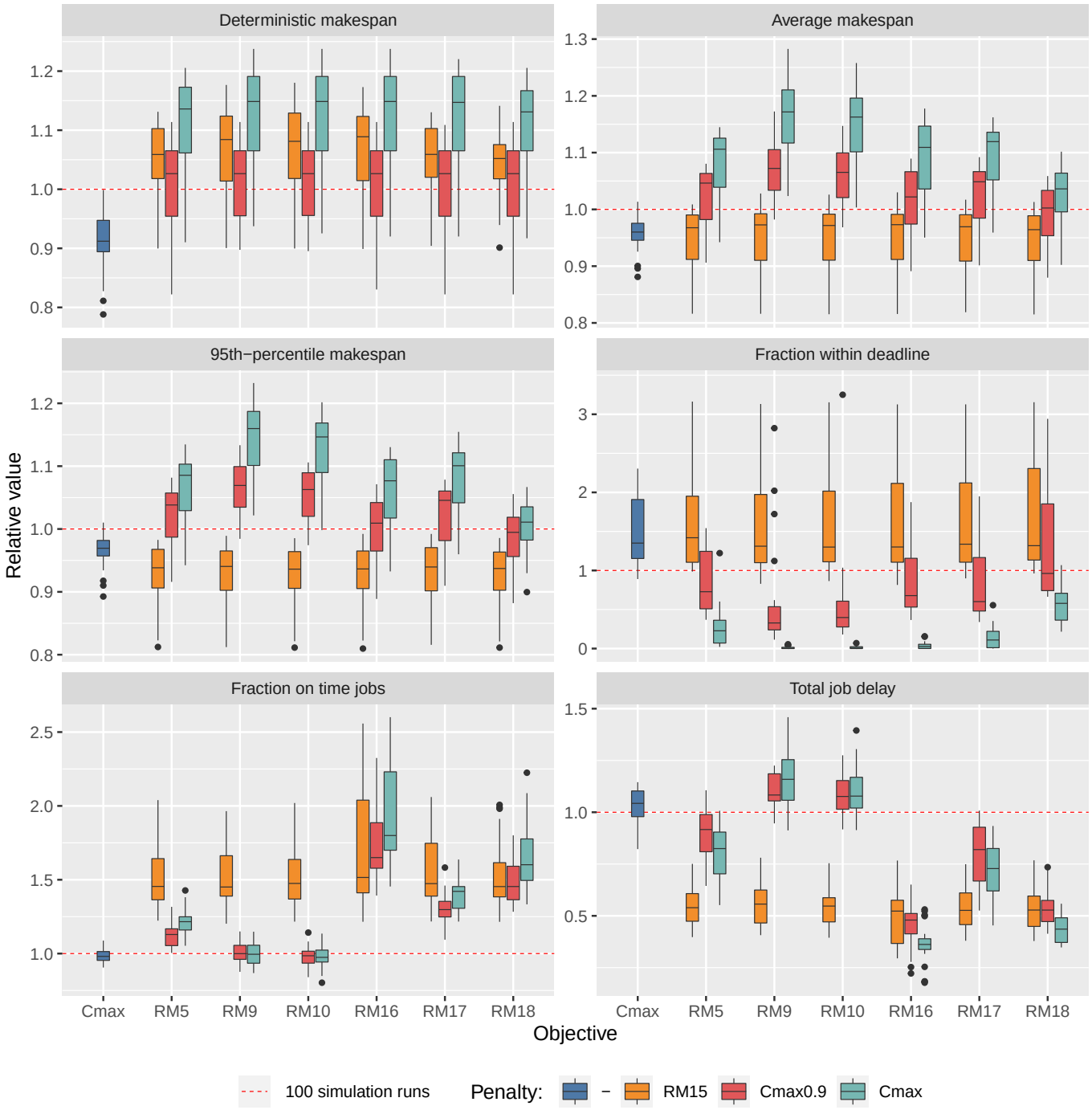

Figure 5.4: Results for distribution LN_{50}. Values are relative to the results of 100 simulation runs in the objective function.

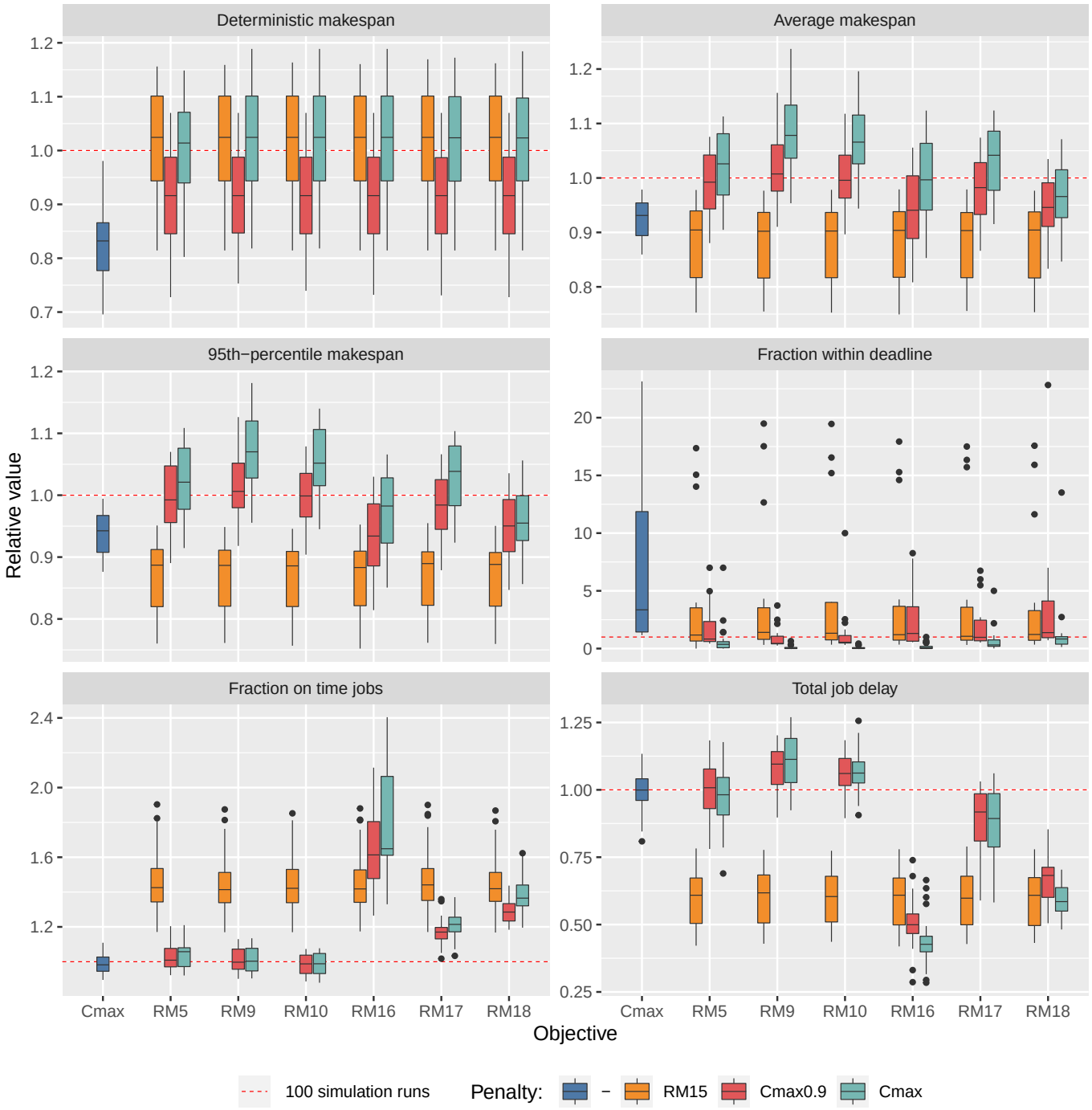

Figure 5.5: Results for distribution *Exp*. Values are relative to the results of 100 simulation runs in the objective function.

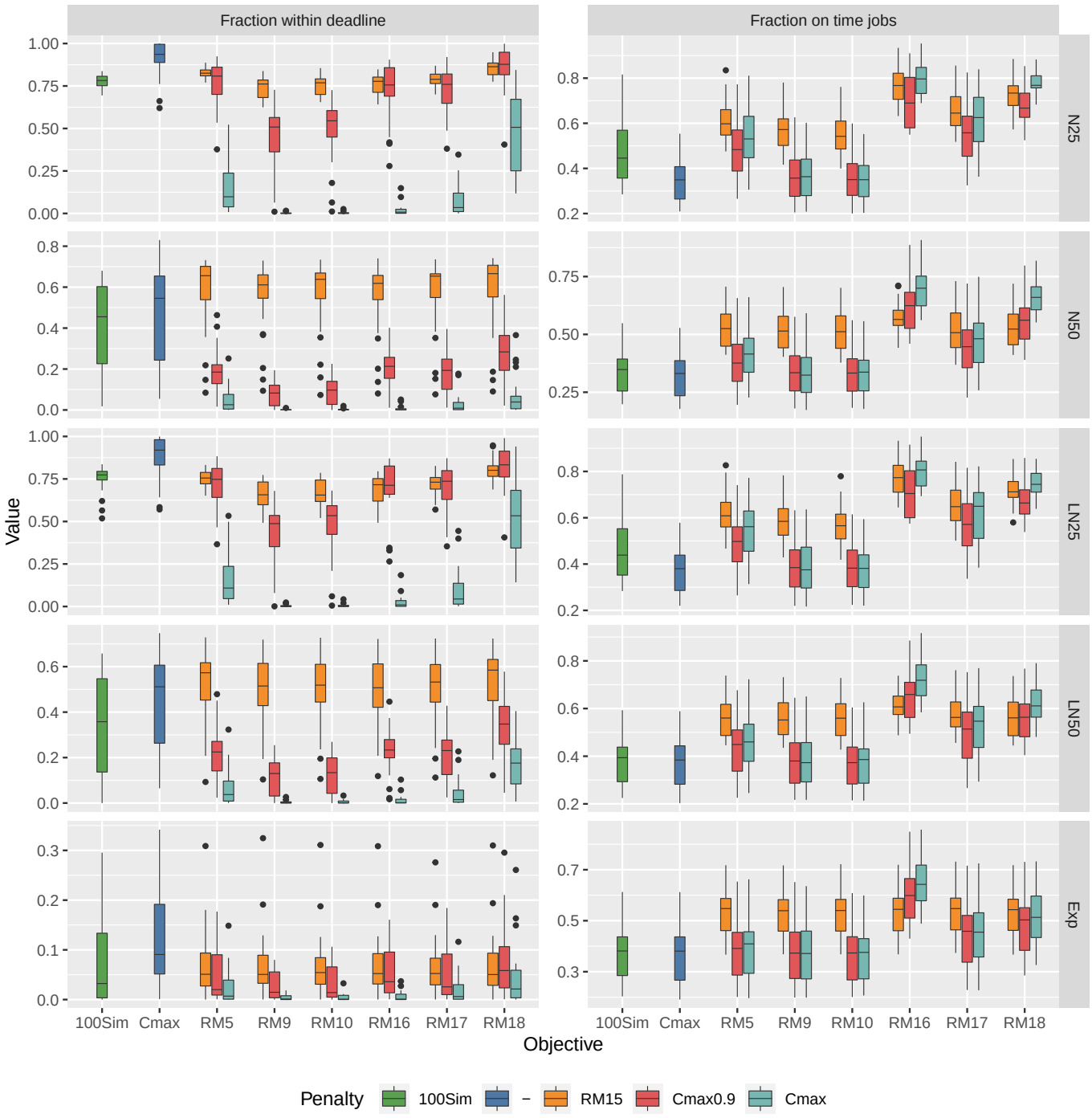

Figure 5.6: Absolute values for fraction within deadline and fraction on time jobs for all distributions and objective functions.

5.4 Conclusion

In this chapter we presented a local search procedure which allowed us to test the performance of the robustness measures in a practical setting. We compared several combinations of robustness measure based objectives and penalties to deterministic makespan minimization and a simulation based objective function.

The results showed that the schedule with the lowest deterministic makespan will not necessarily have the lowest average or 95th percentile makespan or the highest fraction on time jobs. This highlights the importance of taking into account the stochasticity of the job processing times when optimizing quality robustness. Additionally, deterministic makespan minimization will result in a low solution robustness, as it create schedules without buffers.

Furthermore, the results showed the difficulty in using the deterministic makespan as a lower bound for quality robustness. As illustrated, restricting the deterministic makespan to the deadline does not mean the realized makespan will also be within the deadline. In fact, this results in a poor fraction within deadline. This is as expected, since the jobs are likely to take longer than their deterministic processing time. Constraining the makespan to a lower percentage of the deadline will increase the quality robustness, but the difficulty on how to choose this percentage remains.

Subsequently, the results showed the advantages of the normal approximation methods. Not only does using RM_{15} as a penalty result in a high quality robustness, it also increases solution robustness, despite the fact that they may be conflicting objectives. It is likely that penalizing a high expected makespan with RM_{15} results in more stable schedules overall, leading to a better fraction on time jobs and lower total job delay. For high processing time variability, RM_{15} even outperforms 100 simulation runs in both quality and solution robustness, regardless of the solution robustness objective. Additionally, optimizing normal approximation measure RM_{16} results in the most solution robust schedules overall. Alongside the improved robustness, the normal approximation has a much lower computation time then 100 simulation runs, as shown in Section 4.3.2. Therefore, optimizing RM_{16} combined with RM_{15} as a penalty results in a superior objective function, but this requires knowledge about the variance of the processing times.

Next to normal approximation objective RM_{16}, slack-based measures RM_5, RM_{17} and RM_{18} also take into account the processing time variability. The results showed that using these measures as an objective resulted in solution robust schedules, outperforming deterministic makespan minimization and generally the simulation as well. It is as expected that optimizing these measures will result in a better solution robustness than deterministic makespan minimization, as the measures encourage to increase the amount of free slack by inserting buffers.

RM_9 and RM_{10} have the advantage that they do not need information about the processing time distribution. However, they can result in poor schedules when combined with makespan-based penalties. The bad performance can be explained by the combination of our algorithm and the aim of these measures to assign large free slack to jobs with a lot of successors (and a large processing time). Our algorithm first assigns jobs to machines, and then inserts buffer. RM_9 and RM_{10} encourage the increase of free slack for certain jobs in both steps of the algorithm. Increasing free slack in an earliest start schedule without explicitly inserting buffers can only be achieved by assigning the jobs to machines in such a way that the precedence relations cause gaps in the schedule. This results in problematic schedules with an overly complex structure, which is not robust at all. Therefore, RM_9 and RM_{10} do not seem suitable in the approach where jobs are first assigned to machines, and buffers are inserted afterwards.

Chapter 6

Conclusion

6.1 Summary

In this thesis we looked at the stochastic parallel machine scheduling problem with precedence constraints, release dates and a deadline. In this problem, the processing times of the jobs are not deterministic, but stochastic variables. Our aim was to investigate how we can efficiently find solution robust schedules bounded by a deadline. Since repeatedly simulating the schedule to accurately approximate the true robustness is computationally expensive, surrogate robustness measures can provide a more efficient alternative.

Therefore, we evaluated several robustness measures on their ability to estimate the quality and solution robustness of schedules with buffers and a deadline. To that end, we generated numerous schedules for several instances. Using a multi-start hill-climbing approach we obtained various assignments of jobs to machines. Buffers were inserted into these schedules by repeatedly choosing random buffers from different intervals, while taking into account the schedule deadline. This resulted in the variety of schedules with different solution and quality robustness. We compared the estimates of the robustness measures with the results of a Monte Carlo simulation by reporting their Spearman's rank correlation coefficients for various notions of robustness and several processing time probability distributions.

Furthermore, we implemented a local search algorithm which uses the robustness measures to evaluate the quality of the schedule. This allowed us to investigate the performance of the robustness measures in a practical setting. We applied an iterated local search procedure with a variable neighbourhood descent subroutine to assign jobs to machines, and a hill-climbing subroutine to insert buffers into the earliest start schedules. The highest correlating robustness measures for solution robustness were used as an objective. Penalties based on the highest correlating quality robustness measures were used to set a lower bound on quality robustness. An adaptive penalty function was applied to guide the algorithm towards a feasible solution, while not restricting the search space too much. The results were compared to deterministic makespan minimization and a simulation based objective function.

6.2 Conclusion

The results of our simulation study and the local search experiments have shown the advantages of having information about the variance in processing times. The robustness measures based on the normal approximations of the job processing times showed high correlations with the quality and solution robustness, and mostly outperformed the simulation based objective function in the local search procedure.

Additionally, slack-based measures that are based on the expected increase of the pro-

cessing times showed high correlations with solution robustness as well. In the local search procedure, combining these measures with the normal approximation method resulted in high quality and solution robust schedules.

Furthermore, if no information about the job processing time distribution is known, purely slack-based measures could provide an alternative. Several of these measures showed good correlations with solution robustness, but their behaviour in our local search procedure showed some drawbacks. In a local search approach where the machine assignments and the buffers are changed simultaneously, the performance might be better. Despite the high correlation of the deterministic makespan with quality robustness, it is difficult to include it in a local search approach to obtain quality robust schedules.

Finally, we showed that the robustness measures are significantly more computationally efficient then estimating robustness with simulation. Concluding, we have found that robustness measures can provide a good alternative to simulation for generating solution and quality robust schedules bounded by a deadline.

6.3 Future research

6.3.1 Other sources of uncertainty

In this thesis, we have looked at stochastic processing times. Further research can be done on the quality of robustness measures for estimating robustness when other sources of uncertainty are considered. For example, machine breakdowns or stochastic release dates.

6.3.2 Normal approximation for buffered schedules

The normal approximation method by Passage et al. (2016) was developed for expected makespan minimization and computes the job start and completion time distributions with the assumption they are normally distributed. Having a normal distribution as the job start time indicates that the job can also start earlier than the mean. This makes sense in the case of earliest start schedules, for expected makespan minimization for example. However, we investigated buffered schedules where jobs are not allowed to start earlier than their planned start time. This could be represented by a *truncated* distribution. Using the normal approximation with the assumption that the job start and completion times are normally distributed, can give an underestimation of the makespan distribution in the case of buffered schedules. To address this problem, it could be investigated if the normal approximation method can be used with a truncated normal distribution.

6.3.3 Local search improvements

In our local search approach, we set a lower bound on quality robustness by applying a penalty. Other approaches can be to use a weighted objective function or to generate the Pareto front. Further research can be done to examine how the robustness measures behave in such cases.

Additionally, the buffer allocation can be optimized by running an LP, instead of using the hill-climbing procedure. The resulting schedules with have an optimal start time of the jobs according to the robustness measures. This will make the algorithm less computationally efficient, but it can provide additional insight into the ability of robustness measures to serve as objective functions.

Finally, research can be done with an algorithm that determines the machine assignments and buffer allocation in one step, instead of using a two step approach. This will make the search space more complex, but could provide better solutions in some cases, especially for RM_9 and RM_{10}.

References

Al-Fawzan, M. A., & Haouari, M. (2005). A bi-objective model for robust resource-constrained project scheduling. *International Journal of production economics*, *96*(2), 175–187. doi: 10.1016/j.ijpe.2004.04.002

Bölöni, L., & Marinescu, D. C. (2002). Robust scheduling of metaprograms. *Journal of Scheduling*, *5*(5), 395–412. doi: 10.1002/jos.115

Canon, L.-C., & Jeannot, E. (2009). Evaluation and optimization of the robustness of dag schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, *21*(4), 532–546. doi: 10.1109/TPDS.2009.84

Chtourou, H., & Haouari, M. (2008). A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & industrial engineering*, *55*(1), 183–194. doi: 10.1016/j.cie.2007.11.017

Hazır, Ö., Haouari, M., & Erel, E. (2010). Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research*, *207*(2), 633–643. doi: 10.1016/j.ejor.2010.05.046

Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European journal of operational research*, *156*(3), 550–565. doi: 10.1016/S0377-2217(03)00130-9

Hessey, M. S., van den Akker, J. M., & Hoogeveen, J. A. (2019). *Solving stochastic parallel machine scheduling using a metaheuristic approach with efficient robustness estimation* (Master's thesis). Retrieved from `https://studenttheses.uu.nl/handle/20.500.12932/33652`

Joines, J. A., Houck, C. R., et al. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *International conference on evolutionary computation* (pp. 579–584).

Jorge Leon, V., David Wu, S., & Storer, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE transactions*, *26*(5), 32–43. doi: 10.1080/07408179408966626

Khemakhem, M. A., & Chtourou, H. (2013). Efficient robustness measures for the resource-constrained project scheduling problem. *International Journal of Industrial and Systems Engineering*, *14*(2), 245–267. doi: 10.1504/IJISE.2013.053738

Kobylański, P., & Kuchta, D. (2007). A note on the paper by M. A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, *107*(2), 496–501. doi: 10.1016/j.ijpe.2006.07.012

Lambrechts, O., Demeulemeester, E., & Herroelen, W. (2008). A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, *111*(2), 493–508. doi: 10.1016/j.ijpe.2007.02.003

Liang, Y., Cui, N., Hu, X., & Demeulemeester, E. (2020). The integration of resource allocation and time buffering for bi-objective robust project scheduling. *International Journal of Production Research*, *58*(13), 3839–3854.

Nadarajah, S., & Kotz, S. (2008). Exact distribution of the max/min of two gaussian random variables. *IEEE Transactions on very large scale integration (VLSI) systems*, *16*(2), 210–212. doi: 10.1109/TVLSI.2007.912191

Passage, G., van den Akker, J. M., & Hoogeveen, J. A. (2016). *Combining local search and heuristics for solving robust parallel machine scheduling* (Master's thesis). Retrieved from `https://studenttheses.uu.nl/handle/20.500.12932/22475`

Shestak, V., Smith, J., Siegel, H. J., & Maciejewski, A. A. (2006). A stochastic approach to measuring the robustness of resource allocations in distributed systems. In *2006 international conference on parallel processing (icpp'06)* (pp. 459–470). doi: 10.1109/ICPP.2006.14

Shi, Z., Jeannot, E., & Dongarra, J. J. (2006). Robust task scheduling in non-deterministic heterogeneous computing systems. In *2006 ieee international conference on cluster computing* (pp. 1–10). doi: 10.1109/CLUSTR.2006.311868

Thomas, B. W., & Manni, E. (2014). Scheduled penalty variable neighborhood search. *Computers & operations research*, *52*, 170–180.

van den Akker, M., van Blokland, K., & Hoogeveen, H. (2013). Finding robust solutions for the stochastic job shop scheduling problem by including simulation in local search. In *International symposium on experimental algorithms* (pp. 402–413). doi: 10.1007/978-3-642-38527-8_35

van den Broek, R., Hoogeveen, H., & van den Akker, M. (2018). How to Measure the Robustness of Shunting Plans. In *18th workshop on algorithmic approaches for transportation modelling, optimization, and systems (atmos 2018)* (Vol. 65, pp. 3:1–3:13). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi: 10.4230/OASIcs.ATMOS.2018.3

Van de Vonder, S., Demeulemeester, E., & Herroelen, W. (2008). Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, *189*(3), 723–733.

Van de Vonder, S., Demeulemeester, E., Herroelen*, W., & Leus, R. (2006). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, *44*(2), 215–236.

Wilson, M., Klos, T., Witteveen, C., & Huisman, B. (2014). Flexibility and decoupling in Simple Temporal Networks. *Artificial Intelligence*, *214*, 26–44. doi: https://doi.org/10.1016/j.artint.2014.05.003

Appendix A

RM Evaluation Results

The following tables show the result for the initial robustness measure evaluation described in Section 4.2. Tables A.1, A.2, A.3, A.4 show the correlations with the average makespan, fraction within deadline, fraction on time jobs and total job delay, respectively.

		C_{max}	RM_1	RM_2	RM_3	RM_4	RM_5	RM_6	RM_7	RM_8	RM_9	RM_{10}	RM_{11}	RM_{12}	RM_{13}	RM_{14}	RM_{15}	RM_{16}	RM_{17}	RM_{18}
N_{25}	30J-15R-4M	0.98	−0.96	−0.11	−0.98	0.74	0.79	−0.03	−0.77	−0.93	0.89	0.80	0.73	−0.66	−0.13	−0.59	−0.99	0.83	0.83	−0.84
	30J-30R-4M	0.98	−0.97	−0.18	−0.98	0.74	0.85	−0.03	−0.16	−0.02	0.93	0.96	0.84	−0.76	−0.27	−0.55	−0.97	0.79	0.85	−0.84
	30J-75R-4M	0.80	−0.92	−0.13	−0.80	0.09	−0.25	−0.02	−0.55	−0.12	−0.04	−0.25	−0.35	0.37	0.00	−0.90	−0.98	−0.26	−0.32	0.25
	30J-15R-8M	0.93	−0.92	−0.19	−0.93	0.51	0.50	−0.03	−0.40	−0.91	0.85	0.76	0.29	−0.39	−0.45	−0.76	−0.99	0.71	0.75	−0.72
	30J-30R-8M	0.98	−0.96	0.02	−0.98	0.69	0.78	−0.01	−0.33	−0.83	0.91	0.88	0.57	−0.60	−0.38	−0.68	−0.98	0.90	0.85	−0.87
	30J-75R-8M	0.97	−0.84	0.46	−0.97	0.78	0.92	0.00	0.09	−0.24	0.93	0.90	0.90	−0.90	−0.31	−0.60	−0.98	0.94	0.92	−0.95
	100J-50R-6M	0.99	−0.99	−0.06	−0.99	0.83	0.96	−0.02	−0.45	−0.95	0.99	0.98	0.96	−0.93	−0.03	−0.30	−0.87	0.95	0.96	−0.95
	100J-100R-6M	0.99	−0.98	0.24	−0.99	0.84	0.95	−0.02	−0.48	−0.77	0.97	0.95	0.92	−0.93	−0.23	−0.27	−0.93	0.95	0.94	−0.95
	100J-250R-6M	0.96	−0.99	−0.04	−0.96	0.73	0.79	−0.03	−0.20	−0.73	0.87	0.87	0.78	−0.78	−0.15	−0.36	−0.89	0.81	0.80	−0.79
	100J-50R-12M	0.92	−0.90	−0.59	−0.92	0.48	0.38	−0.04	−0.82	−0.05	0.66	0.63	0.38	−0.36	−0.59	−0.76	−0.98	0.46	0.40	−0.37
	100J-100R-12M	0.98	−0.94	0.41	−0.98	0.81	0.95	−0.01	0.05	−0.71	0.94	0.93	0.93	−0.92	−0.18	−0.38	−0.97	0.97	0.94	−0.96
	100J-250R-12M	0.88	−0.97	0.12	−0.88	0.52	0.56	−0.03	−0.15	−0.21	0.85	0.79	0.56	−0.46	−0.33	−0.62	−0.93	0.56	0.51	−0.51
	Average	0.95	−0.94	0.00	−0.95	0.65	0.68	−0.02	−0.35	−0.54	0.81	0.77	0.63	−0.61	−0.25	−0.56	−0.96	0.72	0.70	−0.71
LN_{25}	30J-15R-4M	0.98	−0.96	−0.11	−0.98	0.74	0.80	−0.03	−0.78	−0.93	0.89	0.80	0.78	−0.72	−0.13	−0.59	−0.99	0.83	0.84	−0.85
	30J-30R-4M	0.98	−0.97	−0.18	−0.98	0.74	0.84	−0.03	−0.16	−0.03	0.93	0.96	0.84	−0.78	−0.27	−0.55	−0.97	0.79	0.86	−0.86
	30J-75R-4M	0.81	−0.93	−0.13	−0.81	0.11	−0.19	−0.02	−0.54	−0.14	−0.01	−0.23	−0.28	0.28	−0.01	−0.90	−0.98	−0.24	−0.30	0.20
	30J-15R-8M	0.93	−0.93	−0.19	−0.93	0.52	0.55	−0.03	−0.41	−0.92	0.86	0.77	0.36	−0.46	−0.46	−0.76	−0.99	0.72	0.75	−0.74
	30J-30R-8M	0.98	−0.96	0.02	−0.98	0.69	0.78	−0.01	−0.33	−0.83	0.91	0.88	0.59	−0.62	−0.38	−0.68	−0.97	0.90	0.86	−0.87
	30J-75R-8M	0.97	−0.85	0.46	−0.97	0.78	0.91	0.00	0.09	−0.24	0.93	0.90	0.88	−0.88	−0.32	−0.60	−0.98	0.94	0.93	−0.95
	100J-50R-6M	0.99	−0.99	−0.06	−0.99	0.83	0.96	−0.02	−0.45	−0.95	0.99	0.98	0.95	−0.94	−0.03	−0.30	−0.87	0.95	0.96	−0.95
	100J-100R-6M	0.99	−0.98	0.24	−0.99	0.84	0.95	−0.02	−0.48	−0.77	0.97	0.95	0.94	−0.95	−0.23	−0.27	−0.93	0.96	0.95	−0.95
	100J-250R-6M	0.97	−0.99	−0.05	−0.97	0.74	0.80	−0.03	−0.20	−0.74	0.87	0.87	0.79	−0.79	−0.15	−0.35	−0.89	0.82	0.80	−0.80
	100J-50R-12M	0.93	−0.91	−0.59	−0.93	0.50	0.40	−0.05	−0.81	−0.07	0.68	0.65	0.41	−0.39	−0.59	−0.75	−0.98	0.48	0.40	−0.40
	100J-100R-12M	0.98	−0.94	0.41	−0.98	0.81	0.94	−0.01	0.04	−0.71	0.94	0.93	0.93	−0.93	−0.17	−0.38	−0.97	0.97	0.95	−0.96
	100J-250R-12M	0.89	−0.97	0.12	−0.89	0.53	0.56	−0.03	−0.15	−0.20	0.85	0.79	0.59	−0.50	−0.33	−0.61	−0.93	0.57	0.53	−0.53
	Average	0.95	−0.95	0.00	−0.95	0.65	0.69	−0.02	−0.35	−0.54	0.82	0.77	0.65	−0.64	−0.26	−0.56	−0.95	0.72	0.71	−0.72
Exp	30J-15R-4M	0.76	−0.89	−0.20	−0.76	0.52	0.47	−0.02	−0.72	−0.80	0.66	0.58	0.53	−0.19	−0.19	−0.66	−0.83	0.54	0.59	−0.53
	30J-30R-4M	0.72	−0.88	−0.34	−0.72	0.46	0.55	−0.02	−0.21	−0.35	0.68	0.72	0.55	−0.29	−0.46	−0.66	−0.78	0.38	0.54	−0.42
	30J-75R-4M	0.49	−0.64	−0.08	−0.49	0.07	−0.29	0.03	−0.48	−0.14	−0.09	−0.24	−0.51	0.55	0.06	−0.62	−0.77	−0.21	−0.30	0.24
	30J-15R-8M	0.73	−0.81	−0.34	−0.73	0.42	0.26	0.00	−0.50	−0.76	0.65	0.57	−0.11	0.07	−0.55	−0.62	−0.80	0.50	0.51	−0.51
	30J-30R-8M	0.89	−0.95	−0.08	−0.89	0.65	0.67	−0.02	−0.48	−0.79	0.85	0.79	0.42	−0.43	−0.42	−0.66	−0.90	0.80	0.74	−0.80
	30J-75R-8M	0.66	−0.85	0.08	−0.66	0.53	0.63	−0.02	−0.36	−0.33	0.67	0.57	0.63	−0.55	−0.61	−0.53	−0.76	0.65	0.56	−0.64
	100J-50R-6M	0.74	−0.81	−0.17	−0.74	0.60	0.66	−0.02	−0.45	−0.74	0.73	0.73	0.73	−0.50	−0.21	−0.56	−0.74	0.62	0.65	−0.61
	100J-100R-6M	0.79	−0.82	0.36	−0.79	0.66	0.73	−0.02	−0.24	−0.60	0.75	0.73	0.73	−0.64	−0.10	−0.40	−0.75	0.71	0.70	−0.71
	100J-250R-6M	0.64	−0.76	−0.30	−0.64	0.41	0.43	−0.02	−0.07	−0.65	0.55	0.60	0.38	−0.34	−0.31	−0.52	−0.66	0.41	0.45	−0.38
	100J-50R-12M	0.69	−0.71	−0.65	−0.69	0.26	0.15	−0.01	−0.76	0.19	0.39	0.38	0.21	−0.15	−0.64	−0.72	−0.80	0.19	0.23	−0.11
	100J-100R-12M	0.81	−0.87	0.34	−0.81	0.69	0.78	−0.02	0.05	−0.76	0.81	0.77	0.72	−0.61	−0.06	−0.37	−0.80	0.81	0.77	−0.80
	100J-250R-12M	0.47	−0.69	−0.29	−0.47	0.14	0.11	−0.02	−0.37	−0.37	0.43	0.39	0.08	0.09	−0.42	−0.61	−0.72	0.10	0.08	−0.04
	Average	0.70	−0.81	−0.14	−0.70	0.45	0.43	−0.01	−0.38	−0.51	0.59	0.55	0.36	−0.25	−0.33	−0.58	−0.78	0.46	0.46	−0.44

Table A.1: Spearman's rank correlation coefficients for average makespan

		C_{max}	RM_1	RM_2	RM_3	RM_4	RM_5	RM_6	RM_7	RM_8	RM_9	RM_{10}	RM_{11}	RM_{12}	RM_{13}	RM_{14}	RM_{15}	RM_{16}	RM_{17}	RM_{18}
N_{25}	30J-15R-4M	−0.94	0.96	0.14	0.94	−0.70	−0.72	0.04	0.78	0.93	−0.84	−0.75	−0.68	0.58	0.17	0.64	0.99	−0.76	−0.77	0.78
	30J-30R-4M	−0.93	0.95	0.23	0.93	−0.67	−0.78	0.04	0.22	0.07	−0.88	−0.91	−0.76	0.67	0.31	0.59	0.99	−0.70	−0.78	0.77
	30J-75R-4M	−0.88	0.94	0.14	0.88	−0.21	0.11	0.05	0.48	0.22	−0.10	0.11	0.29	−0.31	0.06	0.86	0.96	0.10	0.18	−0.10
	30J-15R-8M	−0.95	0.94	0.14	0.95	−0.55	−0.56	0.03	0.34	0.92	−0.89	−0.81	−0.33	0.42	0.41	0.77	1.00	−0.76	−0.79	0.77
	30J-30R-8M	−0.94	0.94	0.01	0.94	−0.66	−0.73	0.03	0.34	0.82	−0.89	−0.85	−0.52	0.55	0.37	0.69	0.99	−0.86	−0.81	0.83
	30J-75R-8M	−0.95	0.84	−0.44	0.95	−0.76	−0.90	0.03	−0.07	0.26	−0.90	−0.88	−0.86	0.87	0.33	0.63	0.97	−0.92	−0.90	0.93
	100J-50R-6M	−0.85	0.87	0.05	0.85	−0.73	−0.80	0.07	0.44	0.82	−0.85	−0.84	−0.80	0.77	0.03	0.38	0.88	−0.78	−0.79	0.78
	100J-100R-6M	−0.91	0.93	−0.19	0.91	−0.77	−0.84	0.05	0.49	0.74	−0.87	−0.85	−0.80	0.82	0.25	0.37	0.93	−0.84	−0.82	0.83
	100J-250R-6M	−0.88	0.95	0.00	0.88	−0.64	−0.66	0.06	0.13	0.65	−0.76	−0.77	−0.65	0.65	0.16	0.44	0.95	−0.69	−0.68	0.66
	100J-50R-12M	−0.93	0.91	0.55	0.93	−0.52	−0.42	0.06	0.77	0.09	−0.68	−0.66	−0.41	0.37	0.56	0.75	1.00	−0.50	−0.43	0.41
	100J-100R-12M	−0.96	0.92	−0.35	0.96	−0.79	−0.90	0.03	0.03	0.67	−0.89	−0.88	−0.90	0.88	0.20	0.45	0.98	−0.93	−0.90	0.93
	100J-250R-12M	−0.86	0.94	−0.08	0.86	−0.52	−0.54	0.06	0.19	0.22	−0.81	−0.75	−0.53	0.43	0.37	0.62	0.96	−0.56	−0.50	0.51
	Average	−0.91	0.92	0.02	0.91	−0.63	−0.64	0.05	0.35	0.53	−0.78	−0.74	−0.58	0.56	0.27	0.60	0.97	−0.68	−0.67	0.68
LN_{25}	30J-15R-4M	−0.95	0.96	0.13	0.95	−0.71	−0.75	0.04	0.79	0.93	−0.86	−0.76	−0.74	0.67	0.16	0.63	1.00	−0.78	−0.79	0.81
	30J-30R-4M	−0.94	0.96	0.22	0.94	−0.69	−0.78	0.04	0.20	0.07	−0.89	−0.92	−0.78	0.70	0.31	0.59	0.99	−0.71	−0.81	0.79
	30J-75R-4M	−0.88	0.94	0.14	0.88	−0.21	0.06	0.05	0.47	0.23	−0.12	0.11	0.22	−0.21	0.06	0.86	0.96	0.10	0.16	−0.06
	30J-15R-8M	−0.95	0.94	0.14	0.95	−0.55	−0.59	0.03	0.34	0.93	−0.90	−0.82	−0.40	0.48	0.40	0.76	1.00	−0.76	−0.79	0.78
	30J-30R-8M	−0.95	0.95	0.01	0.95	−0.67	−0.74	0.02	0.35	0.82	−0.90	−0.86	−0.54	0.58	0.38	0.69	0.98	−0.87	−0.83	0.84
	30J-75R-8M	−0.95	0.86	−0.43	0.95	−0.76	−0.88	0.02	−0.05	0.25	−0.91	−0.87	−0.85	0.85	0.35	0.63	0.98	−0.92	−0.90	0.92
	100J-50R-6M	−0.90	0.92	0.06	0.90	−0.76	−0.84	0.06	0.46	0.87	−0.90	−0.89	−0.84	0.82	0.05	0.40	0.92	−0.82	−0.84	0.83
	100J-100R-6M	−0.94	0.96	−0.21	0.94	−0.80	−0.89	0.05	0.50	0.76	−0.91	−0.89	−0.86	0.88	0.25	0.35	0.95	−0.89	−0.88	0.88
	100J-250R-6M	−0.91	0.97	0.02	0.91	−0.67	−0.71	0.05	0.15	0.69	−0.80	−0.81	−0.69	0.70	0.15	0.41	0.94	−0.73	−0.72	0.71
	100J-50R-12M	−0.94	0.93	0.55	0.94	−0.54	−0.46	0.06	0.76	0.12	−0.71	−0.69	−0.46	0.43	0.55	0.73	0.99	−0.53	−0.45	0.46
	100J-100R-12M	−0.96	0.93	−0.36	0.96	−0.80	−0.91	0.02	0.01	0.67	−0.90	−0.88	−0.91	0.90	0.19	0.44	0.98	−0.94	−0.92	0.93
	100J-250R-12M	−0.88	0.95	−0.09	0.88	−0.54	−0.57	0.06	0.19	0.21	−0.83	−0.77	−0.58	0.50	0.38	0.60	0.96	−0.58	−0.54	0.55
	Average	−0.93	0.94	0.02	0.93	−0.64	−0.67	0.04	0.35	0.55	−0.80	−0.76	−0.62	0.61	0.27	0.59	0.97	−0.70	−0.69	0.70
Exp	30J-15R-4M	−0.93	0.93	0.06	0.93	−0.72	−0.74	0.01	0.69	0.87	−0.88	−0.81	−0.63	0.44	0.07	0.54	0.95	−0.81	−0.82	0.80
	30J-30R-4M	−0.92	0.98	0.23	0.92	−0.68	−0.80	0.02	0.16	0.14	−0.88	−0.92	−0.75	0.58	0.34	0.60	0.92	−0.69	−0.79	0.71
	30J-75R-4M	−0.75	0.74	0.08	0.75	−0.37	−0.05	0.01	0.25	0.30	−0.27	−0.09	0.33	−0.38	0.04	0.61	0.78	−0.13	0.00	0.08
	30J-15R-8M	−0.85	0.87	0.08	0.85	−0.58	−0.50	0.01	0.26	0.85	−0.85	−0.80	−0.07	0.10	0.33	0.67	0.87	−0.74	−0.72	0.74
	30J-30R-8M	−0.95	0.97	−0.02	0.95	−0.71	−0.77	0.01	0.33	0.84	−0.91	−0.88	−0.48	0.50	0.38	0.67	0.92	−0.90	−0.83	0.88
	30J-75R-8M	−0.81	0.84	−0.22	0.81	−0.65	−0.78	0.01	0.19	0.32	−0.80	−0.74	−0.72	0.66	0.49	0.56	0.88	−0.80	−0.72	0.79
	100J-50R-6M	−0.95	0.97	0.08	0.95	−0.81	−0.92	0.03	0.47	0.93	−0.95	−0.95	−0.92	0.81	0.07	0.36	0.88	−0.90	−0.91	0.89
	100J-100R-6M	−0.94	0.93	−0.33	0.94	−0.82	−0.92	0.02	0.36	0.70	−0.93	−0.92	−0.90	0.87	0.15	0.24	0.88	−0.92	−0.90	0.91
	100J-250R-6M	−0.89	0.87	0.13	0.89	−0.74	−0.81	0.02	0.19	0.79	−0.87	−0.88	−0.78	0.76	0.16	0.24	0.71	−0.81	−0.82	0.78
	100J-50R-12M	−0.81	0.82	0.38	0.81	−0.63	−0.62	0.00	0.54	0.32	−0.76	−0.75	−0.53	0.49	0.40	0.38	0.74	−0.67	−0.65	0.62
	100J-100R-12M	−0.92	0.91	−0.44	0.92	−0.78	−0.92	0.02	−0.12	0.81	−0.94	−0.92	−0.81	0.77	0.14	0.27	0.87	−0.94	−0.90	0.93
	100J-250R-12M	−0.83	0.85	−0.08	0.83	−0.56	−0.60	0.05	0.17	0.14	−0.79	−0.75	−0.49	0.39	0.34	0.49	0.78	−0.60	−0.55	0.54
	Average	−0.88	0.89	0.00	0.88	−0.67	−0.70	0.02	0.29	0.58	−0.82	−0.78	−0.56	0.50	0.24	0.47	0.85	−0.74	−0.72	0.72

Table A.2: Spearman's rank correlation coefficients for fraction within deadline

		C_{max}	RM_1	RM_2	RM_3	RM_4	RM_5	RM_6	RM_7	RM_8	RM_9	RM_{10}	RM_{11}	RM_{12}	RM_{13}	RM_{14}	RM_{15}	RM_{16}	RM_{17}	RM_{18}
N_{25}	30J-15R-4M	0.89	−0.73	0.12	−0.89	0.84	0.96	0.02	−0.46	−0.75	0.92	0.89	0.79	−0.90	0.08	−0.15	−0.80	1.00	0.96	−0.98
	30J-30R-4M	0.90	−0.76	0.10	−0.90	0.83	0.95	0.02	−0.02	0.32	0.92	0.92	0.92	−0.95	0.08	−0.17	−0.75	0.99	0.97	−0.98
	30J-75R-4M	0.33	0.01	0.14	−0.33	0.78	0.95	0.04	0.59	−0.31	0.72	0.81	0.74	−0.76	−0.11	0.37	0.23	0.97	0.88	−0.97
	30J-15R-8M	0.86	−0.75	0.32	−0.86	0.70	0.87	0.03	0.13	−0.84	0.95	0.95	0.68	−0.74	0.03	−0.49	−0.80	0.98	0.97	−0.98
	30J-30R-8M	0.91	−0.85	0.27	−0.91	0.80	0.91	0.02	−0.11	−0.72	0.89	0.86	0.75	−0.78	−0.19	−0.42	−0.84	0.96	0.96	−0.97
	30J-75R-8M	0.98	−0.82	0.54	−0.98	0.81	0.95	0.01	0.20	−0.20	0.96	0.94	0.92	−0.93	−0.26	−0.58	−0.93	0.97	0.96	−0.98
	100J-50R-6M	0.97	−0.94	0.01	−0.97	0.86	0.99	0.01	−0.41	−0.92	0.97	0.97	0.97	−0.99	0.06	−0.08	−0.76	1.00	0.99	−0.99
	100J-100R-6M	0.97	−0.92	0.32	−0.97	0.87	0.99	0.01	−0.41	−0.69	0.99	0.98	0.98	−0.99	−0.13	−0.04	−0.85	1.00	0.99	−0.99
	100J-250R-6M	0.91	−0.77	0.03	−0.91	0.86	0.99	0.02	−0.22	−0.75	0.96	0.93	0.99	−0.98	0.01	0.06	−0.56	1.00	0.98	−0.99
	100J-50R-12M	0.71	−0.72	−0.06	−0.71	0.84	0.99	0.03	−0.22	−0.72	0.95	0.96	0.93	−0.93	−0.09	0.09	−0.49	1.00	0.98	−0.99
	100J-100R-12M	0.97	−0.91	0.43	−0.97	0.84	0.99	0.01	0.08	−0.74	0.98	0.97	0.94	−0.97	−0.16	−0.23	−0.92	0.99	0.98	−0.99
	100J-250R-12M	0.85	−0.61	0.33	−0.85	0.85	0.99	0.02	−0.04	0.16	0.84	0.78	0.93	−0.94	−0.24	0.03	−0.39	1.00	0.97	−0.99
	Average	0.85	−0.73	0.21	−0.85	0.82	0.96	0.02	−0.07	−0.51	0.92	0.91	0.88	−0.91	−0.08	−0.13	−0.66	0.99	0.97	−0.98
LN_{25}	30J-15R-4M	0.89	−0.72	0.12	−0.89	0.84	0.95	0.02	−0.45	−0.74	0.92	0.89	0.81	−0.93	0.09	−0.14	−0.79	1.00	0.97	−0.98
	30J-30R-4M	0.90	−0.76	0.10	−0.90	0.83	0.94	0.02	−0.02	0.33	0.92	0.91	0.89	−0.94	0.08	−0.17	−0.74	0.99	0.97	−0.97
	30J-75R-4M	0.34	0.00	0.13	−0.34	0.79	0.96	0.04	0.58	−0.31	0.72	0.80	0.80	−0.84	−0.12	0.36	0.22	0.96	0.90	−0.97
	30J-15R-8M	0.86	−0.75	0.31	−0.86	0.70	0.87	0.03	0.12	−0.84	0.96	0.95	0.71	−0.77	0.03	−0.50	−0.81	0.98	0.97	−0.97
	30J-30R-8M	0.91	−0.85	0.28	−0.91	0.80	0.90	0.02	−0.11	−0.72	0.89	0.87	0.74	−0.78	−0.19	−0.43	−0.84	0.96	0.96	−0.96
	30J-75R-8M	0.98	−0.82	0.54	−0.98	0.81	0.94	0.01	0.21	−0.20	0.96	0.95	0.90	−0.91	−0.25	−0.58	−0.93	0.97	0.96	−0.97
	100J-50R-6M	0.97	−0.94	0.01	−0.97	0.86	0.98	0.01	−0.41	−0.92	0.97	0.97	0.95	−0.99	0.07	−0.07	−0.76	1.00	0.99	−0.99
	100J-100R-6M	0.97	−0.92	0.32	−0.97	0.87	0.99	0.01	−0.41	−0.69	0.99	0.98	0.98	−0.99	−0.13	−0.04	−0.85	1.00	0.99	−0.99
	100J-250R-6M	0.91	−0.77	0.03	−0.91	0.86	0.99	0.02	−0.23	−0.75	0.96	0.93	0.99	−0.99	0.01	0.06	−0.56	1.00	0.98	−0.99
	100J-50R-12M	0.71	−0.72	−0.05	−0.71	0.84	0.99	0.03	−0.22	−0.72	0.95	0.96	0.95	−0.95	−0.09	0.08	−0.49	1.00	0.98	−0.99
	100J-100R-12M	0.97	−0.91	0.44	−0.97	0.84	0.98	0.01	0.09	−0.74	0.98	0.97	0.94	−0.97	−0.16	−0.23	−0.92	0.99	0.99	−0.99
	100J-250R-12M	0.85	−0.61	0.34	−0.85	0.85	0.99	0.02	−0.03	0.17	0.84	0.79	0.96	−0.97	−0.24	0.02	−0.39	1.00	0.98	−0.99
	Average	0.85	−0.73	0.21	−0.85	0.82	0.95	0.02	−0.07	−0.51	0.92	0.91	0.88	−0.92	−0.08	−0.14	−0.66	0.99	0.97	−0.98
Exp	30J-15R-4M	0.76	−0.54	0.22	−0.76	0.76	0.87	0.01	−0.34	−0.60	0.79	0.75	0.47	−0.77	0.18	0.02	−0.66	0.93	0.84	−0.92
	30J-30R-4M	0.77	−0.57	0.19	−0.77	0.74	0.85	0.01	0.01	0.44	0.80	0.77	0.77	−0.88	0.21	−0.05	−0.60	0.94	0.84	−0.92
	30J-75R-4M	0.43	−0.24	−0.14	−0.43	0.64	0.64	0.01	0.28	−0.64	0.74	0.69	0.21	−0.29	−0.37	0.13	−0.07	0.80	0.75	−0.70
	30J-15R-8M	0.85	−0.72	0.31	−0.85	0.65	0.73	0.02	0.05	−0.82	0.92	0.87	0.31	−0.45	0.04	−0.50	−0.81	0.94	0.91	−0.94
	30J-30R-8M	0.90	−0.83	0.34	−0.90	0.75	0.85	0.02	−0.13	−0.71	0.92	0.90	0.60	−0.67	−0.14	−0.47	−0.81	0.96	0.90	−0.93
	30J-75R-8M	0.95	−0.76	0.53	−0.95	0.80	0.95	0.01	0.22	−0.23	0.93	0.91	0.74	−0.78	−0.20	−0.52	−0.92	0.96	0.89	−0.96
	100J-50R-6M	0.90	−0.86	0.12	−0.90	0.81	0.94	0.01	−0.36	−0.85	0.92	0.91	0.86	−0.97	0.19	0.06	−0.71	0.96	0.94	−0.95
	100J-100R-6M	0.94	−0.91	0.27	−0.94	0.84	0.95	0.01	−0.46	−0.66	0.97	0.95	0.92	−0.97	−0.15	−0.03	−0.84	0.97	0.95	−0.96
	100J-250R-6M	0.90	−0.81	0.08	−0.90	0.82	0.95	0.01	−0.24	−0.72	0.95	0.91	0.94	−0.95	0.05	0.07	−0.61	0.97	0.95	−0.94
	100J-50R-12M	0.73	−0.73	−0.05	−0.73	0.83	0.94	0.02	−0.25	−0.67	0.95	0.96	0.76	−0.76	−0.08	0.02	−0.52	0.98	0.94	−0.96
	100J-100R-12M	0.97	−0.90	0.46	−0.97	0.84	0.98	0.01	0.09	−0.74	0.97	0.96	0.84	−0.88	−0.18	−0.26	−0.91	0.99	0.95	−0.98
	100J-250R-12M	0.86	−0.63	0.43	−0.86	0.83	0.95	0.01	0.10	0.11	0.86	0.83	0.83	−0.85	−0.20	−0.03	−0.42	0.94	0.92	−0.91
	Average	0.83	−0.71	0.23	−0.83	0.77	0.88	0.01	−0.08	−0.51	0.89	0.87	0.69	−0.77	−0.05	−0.13	−0.66	0.95	0.90	−0.92

Table A.3: Spearman's rank correlation coefficients for fraction on time jobs

		C_{max}	RM_1	RM_2	RM_3	RM_4	RM_5	RM_6	RM_7	RM_8	RM_9	RM_{10}	RM_{11}	RM_{12}	RM_{13}	RM_{14}	RM_{15}	RM_{16}	RM_{17}	RM_{18}
N_{25}	30J-15R-4M	−0.88	0.71	−0.11	0.88	−0.83	−0.96	−0.02	0.43	0.73	−0.92	−0.90	−0.79	0.89	−0.08	0.13	0.78	−0.99	−0.96	0.98
	30J-30R-4M	−0.90	0.77	−0.09	0.90	−0.83	−0.96	−0.02	0.00	−0.31	−0.92	−0.92	−0.91	0.95	−0.07	0.18	0.75	−0.99	−0.97	0.98
	30J-75R-4M	−0.32	0.00	−0.09	0.32	−0.79	−0.96	−0.04	−0.57	0.30	−0.71	−0.77	−0.75	0.78	0.15	−0.39	−0.24	−0.93	−0.84	0.97
	30J-15R-8M	−0.81	0.70	−0.38	0.81	−0.70	−0.89	−0.03	−0.22	0.79	−0.93	−0.96	−0.72	0.76	−0.10	0.44	0.75	−0.97	−0.95	0.98
	30J-30R-8M	−0.91	0.86	−0.21	0.91	−0.79	−0.91	−0.02	0.11	0.72	−0.89	−0.88	−0.72	0.75	0.24	0.46	0.85	−0.96	−0.95	0.97
	30J-75R-8M	−0.98	0.82	−0.54	0.98	−0.81	−0.96	−0.01	−0.21	0.20	−0.96	−0.95	−0.92	0.93	0.26	0.58	0.93	−0.97	−0.96	0.98
	100J-50R-6M	−0.97	0.94	−0.01	0.97	−0.85	−0.99	−0.01	0.40	0.92	−0.97	−0.97	−0.97	0.99	−0.07	0.07	0.76	−1.00	−0.99	0.99
	100J-100R-6M	−0.97	0.92	−0.31	0.97	−0.87	−0.99	−0.01	0.40	0.69	−0.99	−0.98	−0.98	0.98	0.14	0.04	0.85	−1.00	−0.99	0.99
	100J-250R-6M	−0.91	0.77	−0.02	0.91	−0.86	−0.99	−0.02	0.22	0.75	−0.96	−0.93	−0.98	0.98	0.00	−0.05	0.55	−1.00	−0.98	0.99
	100J-50R-12M	−0.68	0.69	0.03	0.68	−0.83	−0.99	−0.03	0.18	0.73	−0.93	−0.95	−0.93	0.93	0.06	−0.12	0.45	−0.99	−0.98	0.99
	100J-100R-12M	−0.97	0.91	−0.44	0.97	−0.84	−0.99	−0.01	−0.09	0.74	−0.98	−0.97	−0.94	0.97	0.16	0.24	0.93	−0.99	−0.98	0.99
	100J-250R-12M	−0.83	0.59	−0.34	0.83	−0.85	−0.98	−0.02	0.04	−0.22	−0.81	−0.77	−0.91	0.92	0.21	−0.02	0.36	−0.99	−0.96	0.99
	Average	−0.84	0.72	−0.21	0.84	−0.82	−0.96	−0.02	0.06	0.50	−0.91	−0.91	−0.88	0.90	0.08	0.13	0.64	−0.98	−0.96	0.98
LN_{25}	30J-15R-4M	−0.88	0.70	−0.13	0.88	−0.83	−0.95	−0.02	0.42	0.72	−0.91	−0.89	−0.81	0.93	−0.09	0.12	0.77	−0.99	−0.96	0.98
	30J-30R-4M	−0.90	0.75	−0.10	0.90	−0.83	−0.94	−0.02	−0.01	−0.32	−0.91	−0.91	−0.88	0.94	−0.08	0.17	0.74	−0.99	−0.96	0.97
	30J-75R-4M	−0.34	0.01	−0.07	0.34	−0.78	−0.96	−0.04	−0.55	0.31	−0.70	−0.76	−0.81	0.84	0.17	−0.38	−0.24	−0.92	−0.87	0.96
	30J-15R-8M	−0.82	0.69	−0.38	0.82	−0.70	−0.88	−0.03	−0.22	0.79	−0.93	−0.96	−0.74	0.78	−0.10	0.44	0.75	−0.97	−0.95	0.97
	30J-30R-8M	−0.91	0.86	−0.21	0.91	−0.79	−0.89	−0.02	0.11	0.72	−0.89	−0.88	−0.71	0.76	0.24	0.47	0.86	−0.96	−0.95	0.96
	30J-75R-8M	−0.98	0.82	−0.54	0.98	−0.81	−0.94	−0.01	−0.21	0.20	−0.96	−0.95	−0.90	0.91	0.25	0.58	0.93	−0.97	−0.96	0.97
	100J-50R-6M	−0.96	0.94	−0.02	0.96	−0.85	−0.98	−0.01	0.40	0.92	−0.97	−0.97	−0.95	0.98	−0.08	0.07	0.75	−1.00	−0.99	0.99
	100J-100R-6M	−0.97	0.92	−0.30	0.97	−0.87	−0.99	−0.01	0.40	0.69	−0.99	−0.98	−0.98	0.99	0.14	0.04	0.85	−1.00	−0.99	0.99
	100J-250R-6M	−0.91	0.77	−0.02	0.91	−0.86	−0.99	−0.02	0.23	0.75	−0.96	−0.93	−0.98	0.99	0.01	−0.05	0.56	−1.00	−0.98	0.98
	100J-50R-12M	−0.68	0.69	0.03	0.68	−0.83	−0.98	−0.03	0.18	0.73	−0.94	−0.95	−0.94	0.95	0.06	−0.11	0.46	−0.99	−0.98	0.99
	100J-100R-12M	−0.97	0.91	−0.45	0.97	−0.84	−0.97	−0.01	−0.09	0.74	−0.98	−0.97	−0.94	0.97	0.17	0.23	0.92	−0.99	−0.98	0.99
	100J-250R-12M	−0.83	0.59	−0.35	0.83	−0.85	−0.98	−0.02	0.03	−0.23	−0.82	−0.77	−0.94	0.95	0.20	−0.02	0.36	−0.99	−0.97	0.99
	Average	−0.85	0.72	−0.21	0.85	−0.82	−0.95	−0.02	0.06	0.50	−0.91	−0.91	−0.88	0.91	0.07	0.13	0.64	−0.98	−0.96	0.98
Exp	30J-15R-4M	−0.64	0.39	−0.24	0.64	−0.68	−0.81	−0.01	0.19	0.46	−0.70	−0.68	−0.39	0.74	−0.19	−0.12	0.53	−0.85	−0.74	0.85
	30J-30R-4M	−0.69	0.49	−0.21	0.69	−0.68	−0.79	−0.01	−0.10	−0.46	−0.70	−0.69	−0.70	0.85	−0.20	−0.01	0.51	−0.89	−0.77	0.87
	30J-75R-4M	−0.42	0.23	0.25	0.42	−0.59	−0.59	0.00	−0.21	0.56	−0.64	−0.57	−0.25	0.35	0.47	−0.15	0.01	−0.71	−0.67	0.66
	30J-15R-8M	−0.73	0.56	−0.44	0.73	−0.61	−0.76	−0.01	−0.27	0.71	−0.84	−0.84	−0.44	0.53	−0.19	0.35	0.68	−0.90	−0.86	0.93
	30J-30R-8M	−0.89	0.83	−0.21	0.89	−0.72	−0.83	−0.02	0.12	0.72	−0.88	−0.89	−0.55	0.62	0.24	0.51	0.82	−0.95	−0.87	0.92
	30J-75R-8M	−0.91	0.68	−0.53	0.91	−0.76	−0.90	0.00	−0.26	0.20	−0.89	−0.90	−0.71	0.75	0.14	0.48	0.89	−0.91	−0.85	0.92
	100J-50R-6M	−0.85	0.80	−0.16	0.85	−0.76	−0.90	0.00	0.30	0.80	−0.87	−0.86	−0.80	0.95	−0.24	−0.12	0.66	−0.92	−0.90	0.92
	100J-100R-6M	−0.91	0.88	−0.21	0.91	−0.81	−0.93	−0.01	0.44	0.67	−0.94	−0.94	−0.89	0.95	0.20	0.02	0.81	−0.95	−0.93	0.94
	100J-250R-6M	−0.88	0.78	−0.07	0.88	−0.80	−0.92	0.00	0.26	0.67	−0.91	−0.88	−0.92	0.94	−0.04	−0.06	0.59	−0.95	−0.92	0.91
	100J-50R-12M	−0.68	0.66	0.02	0.68	−0.79	−0.90	−0.01	0.21	0.68	−0.91	−0.91	−0.70	0.71	0.04	−0.05	0.47	−0.94	−0.88	0.94
	100J-100R-12M	−0.95	0.87	−0.46	0.95	−0.82	−0.96	−0.01	−0.10	0.71	−0.95	−0.94	−0.83	0.88	0.21	0.25	0.89	−0.97	−0.93	0.97
	100J-250R-12M	−0.78	0.53	−0.43	0.78	−0.82	−0.94	−0.02	−0.07	−0.24	−0.79	−0.75	−0.78	0.83	0.11	−0.06	0.29	−0.95	−0.89	0.95
	Average	−0.78	0.64	−0.22	0.78	−0.74	−0.85	−0.01	0.04	0.46	−0.84	−0.82	−0.66	0.76	0.05	0.09	0.60	−0.91	−0.85	0.90

Table A.4: Spearman's rank correlation coefficients for total job delay

Appendix B

Local Search Results

Tables B.1 and B.2 show the results of the local search experiments described in Chapter 5 for instances 30J-15R-4M and 100J-250R-12M, respectively.

objective:	$100Sim$	$Cmax$	RM_5			RM_9			RM_{10}			RM_{16}			RM_{17}			RM_{18}		
penalty:	$100Sim$	-	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}
N_{25}																				
Deterministic Cmax	85.30	73.00	92.60	84.00	86.30	94.00	84.00	90.80	94.00	84.00	91.00	94.00	84.00	89.10	94.00	84.00	87.70	92.50	84.00	86.30
Average Cmax	90.81	82.74	97.26	91.04	90.14	104.30	94.29	92.80	103.73	94.24	92.77	100.49	92.59	92.13	99.20	91.55	91.13	94.97	89.48	90.02
SD Cmax	4.49	4.75	3.98	4.68	3.79	5.34	5.20	2.88	5.33	5.22	2.70	4.21	4.45	3.27	4.43	4.76	3.55	3.91	3.81	3.68
95% Cmax	98.19	90.55	103.81	98.75	96.38	113.08	102.84	97.54	112.49	102.84	97.21	107.41	99.91	97.51	106.48	99.39	96.97	101.41	95.74	96.07
Within deadline (%)	0.78	0.98	0.21	0.76	0.85	0.00	0.51	0.77	0.01	0.53	0.79	0.01	0.66	0.77	0.10	0.73	0.81	0.43	0.88	0.87
On time jobs (%)	0.55	0.38	0.62	0.50	0.57	0.39	0.39	0.58	0.39	0.37	0.56	0.80	0.67	0.72	0.68	0.60	0.67	0.81	0.71	0.71
Total job delay	48.78	75.76	28.16	49.85	35.16	90.87	84.96	43.22	83.75	88.20	41.90	20.58	39.33	27.80	30.51	49.26	29.04	11.95	17.00	16.36
N_{50}																				
Deterministic Cmax	77.80	73.20	91.10	84.00	88.40	94.00	84.00	88.90	94.00	84.00	88.60	94.00	84.00	88.70	93.30	84.00	88.70	94.00	84.00	88.70
Average Cmax	95.01	93.78	106.62	102.39	93.83	115.79	105.27	94.00	114.88	105.20	93.85	108.47	101.83	93.92	109.37	102.14	93.83	105.08	99.97	93.84
SD Cmax	9.63	9.65	9.59	9.94	7.01	11.23	10.41	6.78	11.01	10.36	6.88	8.29	9.02	6.77	10.02	10.02	6.80	7.41	8.51	6.74
95% Cmax	110.84	109.65	122.39	118.74	105.36	134.27	122.39	105.16	132.99	122.25	105.16	122.11	116.67	105.06	125.85	118.63	105.02	117.26	113.96	104.93
Within deadline (%)	0.49	0.54	0.07	0.21	0.64	0.00	0.13	0.64	0.00	0.14	0.64	0.00	0.20	0.64	0.04	0.22	0.64	0.05	0.25	0.64
On time jobs (%)	0.37	0.36	0.45	0.42	0.51	0.36	0.37	0.51	0.35	0.36	0.50	0.74	0.63	0.51	0.53	0.48	0.51	0.71	0.59	0.51
Total job delay	156.95	163.92	122.82	140.16	114.71	199.34	176.39	116.43	188.36	177.93	119.30	53.69	87.53	115.86	116.52	134.46	117.15	35.86	62.54	116.66
LN_{25}																				
Deterministic Cmax	83.80	73.10	92.50	84.00	85.80	94.00	84.00	90.50	94.00	84.00	90.80	94.00	84.00	89.10	93.60	84.00	87.80	91.60	84.00	86.00
Average Cmax	90.19	83.73	97.79	91.25	90.74	104.43	94.91	93.23	104.33	94.69	93.25	100.87	92.86	92.64	98.58	91.57	91.94	94.98	89.88	90.23
SD Cmax	5.53	5.81	5.03	5.57	4.96	6.21	6.21	3.82	6.29	6.11	3.61	4.96	5.31	4.21	5.05	5.70	4.48	4.77	4.93	4.75
95% Cmax	99.29	93.28	106.07	100.42	98.90	114.65	105.12	99.52	114.67	104.74	99.18	109.03	101.59	99.56	106.90	100.94	99.32	102.81	97.98	98.05
Within deadline (%)	0.78	0.95	0.24	0.74	0.78	0.01	0.49	0.71	0.01	0.51	0.73	0.02	0.64	0.72	0.19	0.72	0.74	0.46	0.82	0.81
On time jobs (%)	0.56	0.40	0.62	0.54	0.60	0.43	0.41	0.58	0.40	0.40	0.57	0.82	0.69	0.73	0.71	0.60	0.68	0.80	0.71	0.73
Total job delay	53.40	81.46	33.70	48.37	37.11	86.39	85.66	46.08	91.67	85.76	43.44	20.63	38.14	29.63	29.76	50.36	34.88	16.46	23.27	20.44
LN_{50}																				
Deterministic Cmax	78.90	72.90	92.00	84.00	88.70	94.00	84.00	88.70	94.00	84.00	89.30	94.00	84.00	89.00	94.00	84.00	88.60	92.50	84.00	88.50
Average Cmax	97.44	94.96	108.41	103.19	95.97	115.76	106.06	95.83	115.05	106.74	95.93	110.49	103.39	95.98	110.73	103.53	95.73	104.54	100.93	95.68
SD Cmax	13.74	13.83	13.57	13.47	10.63	14.86	14.26	10.56	14.40	14.52	9.98	12.49	12.95	10.40	13.66	14.13	10.27	11.96	12.96	10.52
95% Cmax	120.03	117.70	130.74	125.35	113.45	140.21	129.52	113.20	138.73	130.62	112.35	131.03	124.69	113.08	133.20	126.77	112.63	124.21	122.24	112.99
Within deadline (%)	0.47	0.54	0.11	0.27	0.60	0.01	0.20	0.61	0.01	0.19	0.60	0.01	0.25	0.61	0.05	0.28	0.60	0.18	0.34	0.61
On time jobs (%)	0.43	0.42	0.53	0.49	0.56	0.44	0.43	0.56	0.41	0.41	0.57	0.77	0.67	0.57	0.60	0.55	0.56	0.71	0.63	0.56
Total job delay	164.10	162.68	115.52	134.57	120.13	180.99	172.80	120.10	181.17	188.91	114.67	59.87	86.38	117.10	110.45	127.55	119.64	58.83	74.29	118.64
Exp																				
Deterministic Cmax	84.50	73.20	91.20	84.00	94.00	94.00	84.00	94.00	94.00	84.00	94.00	94.00	84.00	94.00	93.90	84.00	94.00	93.90	84.00	94.00
Average Cmax	123.55	119.91	133.43	130.10	116.86	140.29	131.70	117.12	140.24	131.14	117.04	131.33	124.40	117.25	136.15	127.58	117.11	128.85	125.85	116.94
SD Cmax	28.65	29.30	30.56	31.02	23.59	31.38	31.03	24.28	31.60	30.79	24.34	26.68	27.57	24.23	30.94	29.80	24.12	28.06	29.00	24.16
95% Cmax	170.67	168.11	183.69	181.12	155.66	191.90	182.74	157.06	192.21	181.80	157.08	175.21	169.75	157.11	187.04	176.60	156.79	175.01	173.56	156.68
Within deadline (%)	0.13	0.19	0.05	0.09	0.05	0.01	0.07	0.05	0.01	0.07	0.04	0.02	0.10	0.05	0.03	0.10	0.04	0.06	0.10	0.04
On time jobs (%)	0.44	0.43	0.46	0.43	0.57	0.45	0.43	0.57	0.41	0.42	0.57	0.71	0.66	0.57	0.52	0.52	0.58	0.62	0.55	0.57
Total job delay	350.65	337.41	321.14	363.70	245.18	384.48	390.21	253.19	414.69	391.00	248.26	150.91	181.51	247.97	300.46	288.99	247.03	171.79	212.11	248.65

Table B.1: Results for instance 30J-15R-4M. Deadline: 94

objective:	$100Sim$	$Cmax$	RM_5			RM_9			RM_{10}			RM_{16}			RM_{17}			RM_{18}		
penalty:	$100Sim$	-	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}	$Cmax$	$Cmax0.9$	RM_{15}
N_{25}																				
Deterministic Cmax	149.70	144.00	162.90	146.00	158.60	163.00	146.00	159.90	163.00	146.00	159.50	163.00	146.00	157.90	163.00	146.00	158.50	163.00	146.00	156.70
Average Cmax	159.22	156.17	170.65	157.10	160.98	182.36	165.70	162.26	181.01	164.26	161.84	170.66	156.19	160.83	172.16	158.48	161.17	166.79	155.15	160.42
SD Cmax	6.14	6.83	5.79	6.38	3.32	6.85	6.60	3.18	6.91	6.59	3.24	4.77	5.32	3.92	6.00	6.38	3.66	4.00	6.30	4.17
95% Cmax	169.31	167.41	180.17	167.59	166.44	193.63	176.56	167.49	192.38	175.10	167.17	178.50	164.94	167.28	182.03	168.98	167.19	173.38	165.52	167.28
Within deadline (%)	0.75	0.83	0.05	0.82	0.80	0.00	0.37	0.73	0.00	0.46	0.76	0.01	0.89	0.78	0.02	0.77	0.77	0.14	0.88	0.78
On time jobs (%)	0.34	0.28	0.48	0.45	0.60	0.29	0.28	0.53	0.29	0.30	0.52	0.87	0.80	0.83	0.56	0.51	0.61	0.68	0.61	0.65
Total job delay	366.66	445.98	231.50	288.32	126.79	552.50	551.20	180.68	509.50	506.92	170.27	55.57	94.89	61.20	193.25	267.11	146.33	84.03	148.34	96.50
N_{50}																				
Deterministic Cmax	147.30	144.00	162.90	146.00	154.10	163.00	146.00	154.50	163.00	146.00	153.80	163.00	146.00	154.40	163.00	146.00	154.40	163.00	146.00	154.40
Average Cmax	172.71	175.26	186.70	175.57	162.28	204.97	186.45	162.36	201.62	185.09	162.09	185.17	171.61	162.46	191.70	177.75	162.39	176.91	169.29	162.25
SD Cmax	12.78	13.25	13.20	12.95	10.32	13.57	12.64	10.33	13.91	12.93	10.33	10.83	10.67	10.33	13.47	13.25	10.18	10.93	12.37	10.14
95% Cmax	193.74	197.05	208.42	196.86	179.26	227.29	207.24	179.36	224.50	206.36	179.08	202.98	189.16	179.46	213.86	199.55	179.14	194.90	189.63	178.93
Within deadline (%)	0.26	0.18	0.01	0.17	0.65	0.00	0.01	0.66	0.00	0.03	0.65	0.00	0.22	0.65	0.00	0.13	0.65	0.06	0.33	0.65
On time jobs (%)	0.26	0.24	0.36	0.33	0.54	0.25	0.26	0.52	0.27	0.26	0.52	0.75	0.68	0.57	0.38	0.36	0.55	0.56	0.49	0.53
Total job delay	942.45	1046.69	743.33	823.99	345.81	1263.83	1164.94	377.06	1103.61	1133.79	378.92	220.05	303.34	330.96	777.75	850.38	358.45	295.61	437.51	355.72
LN_{25}																				
Deterministic Cmax	147.60	144.00	162.80	146.00	158.40	163.00	146.10	159.10	163.00	146.00	159.90	163.00	146.00	157.90	163.00	146.00	157.90	163.00	146.00	157.30
Average Cmax	158.17	156.34	172.12	158.11	161.71	183.60	166.58	162.92	181.45	165.29	163.27	171.34	157.27	161.61	173.48	159.26	161.63	166.32	154.94	161.27
SD Cmax	7.41	7.70	6.87	7.51	4.67	7.93	7.83	4.49	7.88	7.78	4.38	5.47	6.11	4.79	7.14	7.59	4.96	4.24	6.99	4.84
95% Cmax	170.37	169.00	183.41	170.47	169.39	196.64	179.45	170.31	194.41	178.08	170.47	180.34	167.32	169.49	185.23	171.74	169.79	173.29	166.44	169.23
Within deadline (%)	0.76	0.81	0.04	0.76	0.73	0.00	0.36	0.64	0.00	0.43	0.62	0.01	0.84	0.71	0.02	0.72	0.72	0.18	0.87	0.73
On time jobs (%)	0.35	0.30	0.48	0.46	0.57	0.31	0.31	0.56	0.33	0.31	0.54	0.86	0.80	0.83	0.55	0.50	0.60	0.64	0.59	0.62
Total job delay	393.75	451.99	259.50	317.45	175.13	561.89	559.30	187.05	491.28	515.27	192.54	64.99	105.69	70.79	241.54	302.03	179.89	124.45	176.62	141.21
LN_{50}																				
Deterministic Cmax	152.30	144.00	162.80	146.00	154.30	163.00	146.00	154.50	163.00	146.00	154.40	163.00	146.00	154.30	163.00	146.00	154.30	163.00	146.00	154.10
Average Cmax	184.54	178.88	192.85	181.41	165.85	208.27	191.62	165.83	205.20	189.65	165.52	188.56	175.92	165.53	194.65	182.89	165.76	180.42	172.87	165.46
SD Cmax	18.00	18.43	18.39	18.33	13.81	19.40	18.86	13.67	18.59	18.60	13.50	14.92	15.67	13.65	18.54	18.71	13.66	15.80	17.16	13.58
95% Cmax	214.15	209.20	223.10	211.55	188.56	240.18	222.64	188.32	235.78	220.25	187.73	213.10	201.69	187.99	225.16	213.67	188.23	206.40	201.10	187.80
Within deadline (%)	0.11	0.20	0.01	0.15	0.55	0.00	0.03	0.55	0.00	0.04	0.56	0.00	0.20	0.56	0.00	0.13	0.55	0.08	0.31	0.56
On time jobs (%)	0.29	0.29	0.37	0.34	0.56	0.30	0.29	0.54	0.30	0.30	0.54	0.75	0.68	0.59	0.44	0.42	0.56	0.49	0.48	0.56
Total job delay	1065.86	1113.36	936.53	1006.00	441.51	1268.73	1274.95	455.39	1153.74	1184.91	463.32	270.49	363.94	402.67	784.84	896.42	441.62	545.28	611.14	429.32
Exp																				
Deterministic Cmax	172.50	144.00	162.90	146.00	163.00	163.00	146.10	163.00	163.00	146.00	163.00	163.00	146.00	163.00	163.00	146.00	163.00	163.00	146.00	163.00
Average Cmax	254.00	231.08	244.35	234.33	201.78	265.31	248.77	201.57	259.15	244.27	202.10	232.38	218.26	201.79	248.54	234.13	202.25	226.91	219.01	201.98
SD Cmax	40.37	39.16	39.81	38.87	31.03	41.60	39.90	31.31	40.80	39.38	31.52	33.97	33.17	30.78	40.47	39.55	31.36	36.48	36.01	31.14
95% Cmax	320.40	295.49	309.84	298.26	252.81	333.73	314.41	253.07	326.26	309.05	253.96	288.26	272.82	252.41	315.11	299.18	253.84	286.92	278.23	253.20
Within deadline (%)	0.00	0.01	0.00	0.01	0.02	0.00	0.00	0.02	0.00	0.00	0.03	0.00	0.01	0.02	0.00	0.01	0.02	0.01	0.03	0.02
On time jobs (%)	0.27	0.26	0.29	0.29	0.49	0.26	0.27	0.47	0.28	0.27	0.48	0.60	0.55	0.49	0.34	0.32	0.50	0.37	0.36	0.47
Total job delay	2662.47	2714.57	2525.48	2552.33	1158.73	3082.40	3026.87	1189.43	2806.51	2853.47	1191.79	1058.86	1173.55	1148.97	2341.87	2425.86	1138.95	1796.93	1856.32	1204.50

Table B.2: Results for instance 100J-250R-12M. Deadline: 163