MASTER THESIS

Bc. Jakub Pohly

Risk measures in scheduling problems under uncertainty

Department of Probability and Mathematical Statistics

Supervisor of the master thesis:	doc. RNDr. Martin Branda, Ph.D.
Study programme:	Probability, Mathematical Statistics and Econometrics
Study branch:	Probability, Mathematical Statistics and Econometrics

Prague 2024

I would like to thank everyone who supported me while working on this thesis. Specifically I would like to thank doc. RNDr. Martin Branda, Ph.D. for his insight into this topic and his time he devoted to consultations and comments that helped me finish this thesis.

Title: Risk measures in scheduling problems under uncertainty

Author: Bc. Jakub Pohly

Department: Department of Probability and Mathematical Statistics

Supervisor: doc. RNDr. Martin Branda, Ph.D., Department of Probability and Mathematical Statistics

Abstract: In the presented work we deal with fixed interval scheduling problem with random delays. We present common formulations of the problem and introduce new ones. The aim is introduction of formulations where the actual cost or penalty is properly expressed and also risk of the schedule is taken into account. The main topic of the work is combining existing formulations for FIS problem with risk measures and creating mixed integer linear formulations of them. The new formulations are minimizing expected number of unprocessed jobs which is better linked to penalties than number of overlaps. For formulations based on risk measures we presented mean-variance optimization of number of overlaps and CVaR optimization of both number of overlaps and number of unprocessed jobs. All of the new formulations were reformulated as mixed integer linear problem. Finally we show a numerical study where we implemented two of the new formulations we presented in this work.

Keywords: fixed interval scheduling, risk measures, graph coloring, mixed integer programming, linear programming

Contents

Introduction

In this work we will focus on a fixed interval scheduling problem (FIS). We will work with fixed starting and finishing times of each work, but in the process some random delays of finishing times can occur. The delays can cause that the next job will not be processed and therefore it can make the whole schedule infeasible.

One of the goals of FIS is to assign jobs to machines in such way, that the probability of given schedule is feasible is maximized. Alternatively very common variant of FIS is to minimize penalty of the schedule. We can imagine the penalty as fine for not processing certain job or a fee we need to pay for outsourcing that job. Since the cost of the schedule is random variable, most commonly the expected cost is minimized.

In the first chapter we will introduce the FIS problem and common approaches how the problem is solved. We will also present one new metric which can be useful to use instead of commonly used number of overlaps. The new metric is number of unprocessed jobs, which can be better linked to the penalty we need to pay if not all jobs are processed. The number of unprocessed jobs problem is then reformulated as mixed integer linear optimization. In second chapter we will introduce various common risk measures and we will discuss their properties. The third chapter will be dedicated to applying the risk measures to the FIS problem and finding ways to reformulate these problems as mixed integer programming. The new formulations we presented are mean-variance optimization of number of overlaps, CVaR optimization of number of overlaps and CvaR optimization of number of unprocessed jobs. All of those optimizations combine FIS problem with risk measures. We were able to reformulate all of those new formulations as mixed integer linear programming. In the last chapter we will conduct a numerical study where we will implement some of the formulations we introduced in the third chapter of this thesis.

1. Fixed interval scheduling

In the first chapter we will look at some common formulations of fixed interval scheduling problem. FIS is a special instance of scheduling problems. In scheduling problems the aim is generally to assign resources to some processes. In this problem we work with a set of machines and a set of jobs. Firsstly we will look at the deterministic version of this problem. We will assume that we have available C identical machines which are supposed to do J jobs. The set of the machines will be denoted $\mathcal{C} = \{1, \dots, C\}$ and the set of jobs $\mathcal{J} = \{1, \dots, J\}$. In FIS problem the jobs have predefined starting time and duration (or finishing time). The starting time of job j will be denoted s_j and finishing time f_j.

In the non stochastic version of the problem the aim is to assign each job to one of the machines in such way that each machine can not process more than one job at each time. The optimization in this case can be done by finding the smallest number of machines needed to still having a feasible schedule. The formulation of this problem is:

$$\begin{aligned}
\min_{x,\,y} \quad & \sum_{c \in \mathcal{C}} y_c \\
\text{s.t.} \quad & x_{jc} \le y_c && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\
& \sum_{c \in \mathcal{C}} x_{jc} = 1 && \forall j \in \mathcal{J}, \\
& \sum_{j: s_j \le t < f_j} x_{jc} \le 1 && \forall c \in \mathcal{C},\ \forall t \in \tilde{\mathcal{T}}, \\
& x_{jc} \in \{0,1\} && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\
& y_c \in \{0,1\} && \forall c \in \mathcal{C}
\end{aligned} \tag{1.1}$$

The binary variable x_{jc} indicates if job j is assigned to machine c. Variable y_c indicates if the machine c is used, or in other words if at least one job is assigned to this machine. The second constrain says that each job has to by assigned to exactly one machine. The third constrain says that at each time, each machine can be processing maximum of one job. We can not check this constrain for all timestamps $t \in \mathcal{T} = [0, T]$ because it would create infinite constrains. It is sufficient to check this constrain only for timestamps which correspond to the starting times $\tilde{\mathcal{T}} = \{s_1, \dots, s_J\}$.

We can interpret this problem as a graph coloring problem where each job represents a vertex and each machine is represented by color, the vertices that are connected are those with overlapping processing times which means that they can not be processed by the same machine. The problem of finding the smallest number of machines to process these jobs is then equivalent to finding the chromatic number of corresponding graph and the final coloring corresponds to the optimal schedule.

1.1 Fixed interval scheduling with random delay

In this thesis we will focus on stochastic version of FIS problem, specifically on Fixed interval scheduling with random delay. In this problem we will again work with the set of machines $\mathcal{C}$ an set of jobs $\mathcal{J}$ and the aim will be to assign jobs to machines. The difference between the non stochastic version of FIS and FIS with random delay is that while the starting times are still fixed, there can occur some delay while doing the job and the finishing times can be delayed. The finishing time of job j will be denoted as

$$f_j(\xi) = f_j^0 + D_j(\xi).$$

The finishing time consist of fixed prescribed finishing time f_j^0 and some non negative random delay $D_j(\xi)$. The random delay is a random variable. We will work with assumption that the probability distribution of delays is known and that probability of job finishing on time is positive and comes from a probability space $(\Xi, \mathcal{A}, \mathcal{P})$. Elementary events from this probability space are denoted by $\xi \in \Xi$. Each job has to start exactly at the predefined time and if the machine which it was assigned for is not available, then there is a penalty. The penalty can be understood for example as lost profit for not fulfilled job or as cost of outsourcing the job.

The FIS problems can be divided generally into two categories. The first one is tactical FIS problem and the second one is operational FIS problem. The aim of the tactical problem is as in the previous chapter to minimize the number of machines required to process the jobs. In the operational FIS problem on the other hand we use all the machines and optimize allocation of the jobs to machines in some way.

1.1.1 Tactical FIS with random delay

In the non stochastic version of FIS we were looking for the smallest number of machines to still find a feasible solution. In case of tactical FIS with random delays we will be looking for the smallest number of machines to still find a feasible solution with prescribed reliability. The formulation for tactical FIS problem is taken from Branda [2018].

$$
\begin{aligned}
\min_{x,\,y} \quad & \sum_{c\in\mathcal{C}} y_c \\
\text{s.t.} \quad & \\
& x_{jc} \le y_c \qquad \forall c \in \mathcal{C},\, \forall j \in \mathcal{J}, \\
& \mathcal{P}\left(\xi \in \Xi : \sum_{j:s_j \le t < f_j(\xi)} x_{jc} \le 1, \forall c \in \mathcal{C},\, \forall t \in \tilde{\mathcal{T}}\right) \ge 1 - \varepsilon, \\
& \sum_{c\in\mathcal{C}} x_{jc} = 1 \qquad \forall j \in \mathcal{J}, \\
& \sum_{j:s_j \le t < f_j^0} x_{jc} \le 1 \qquad \forall c \in \mathcal{C},\, \forall t \in \tilde{\mathcal{T}}, \\
& x_{jc} \in \{0,1\} \quad \forall c \in \mathcal{C},\, \forall j \in \mathcal{J}, \\
& y_c \in \{0,1\} \quad \forall c \in \mathcal{C}
\end{aligned}
\tag{1.2}
$$

The formulation of this problem is similar to formulation (1.1). The difference is that now we will add a new constraint which ensures prescribed level of reliability $1 - \varepsilon$. This new constraint ensures that only a subset of feasible solutions for problem (1.1) is feasible also for (1.2). The difference is that schedules which are not fulfilled completely with probability greater than ε will be unfeasible for this problem.

It is good to point out that while the fourth constraint says that at each time only one job can be scheduled to each machine while taking into account only prescribed finishing times of jobs. In the second constraint we check if each machine has assigned at most one job at the time while taking into account finishing times with delays.

If we consider that no delay would occur with probability 1 then the fourth constraint would ensure that the probability from second constraint will be 1 for all feasible solutions therefore second constraint will be fulfilled. This means that the problem (1.1) is a special case of problem (1.2).

1.2 Operational FIS with random delays

In the rest of the thesis we will discuss the operational FIS problem. The aim of operational FIS problem is to find schedule in which the most jobs are done. Because of the delays there is generally no schedule with reliability 100% and therefore we want to find a schedule which will have the least number of processing time overlaps between jobs scheduled to one machine.

Such optimization problem can be as formulated as two stage optimization:

$$\begin{aligned}
\min_{x,\,y} \quad & \sum_{c\in\mathcal{C}}\sum_{j\in\mathcal{J}} y_{jc}(\xi) \\
\text{s.t.} \quad & \\
& \sum_{c\in\mathcal{C}} x_{jc} = 1 && \forall j \in \mathcal{J}, \\
& \sum_{j:s_j\le t<f_j^0} x_{jc} \le 1 && \forall c \in \mathcal{C},\ \forall t \in \tilde{\mathcal{T}}, \\
& x_{jc} \in \{0,1\} && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\
& \sum_{k:f_j\le s_k<f_j(\xi)} x_{kc} \le y_{jc}(\xi) + J(1-x_{jc}) && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J},\ \forall \xi \in \Xi, \\
& y_{jc}(\xi) \in \mathbb{N} && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J},\ \forall \xi \in \Xi.
\end{aligned} \tag{1.3}$$

The first three constraints are similarly as in problems (1.1) and (1.2) to ensure that each job is scheduled exactly to one machine and that each machine is scheduled maximum one job at each time. The last two constraints introduce a new second stage decision variable $y_{jc}(\xi)$ which is dependent on realizations of delays $\xi \in \Xi$. The variable attains positive values ($y_{jc}(\xi) > 0$) when the job j is scheduled to machine c or ($x_{jc} = 1$) and the delay of job j, $D_j(\xi)$, caused at least one subsequent job on machine c to be not executed. The number $y_{jc}(\xi)$ indicates exactly how many jobs were not executed because of the delay $D_j(\xi)$.

Since the problem (1.3) is stochastic optimization problem it can not be solved directly. Instead we can look at other qualities that can be optimized. For example we can look for the schedule with the highest reliability, lowest expected number of jobs not executed, lowest penalty for not completed jobs and others.

In the following sections we will work with assumption that the number of jobs is not less than number of machines ($J \ge C$) and also we will assume that each machine has at least one job assigned. If some machine has no assigned job and another machine has at least two jobs, we can take one job from second machine and assign it to the first machine and the resulting schedule will not be worse.

1.2.1 Maximizing reliability of schedule

In Branda et al. [2016] two approaches to operational FIS problem were presented, first of them was called *Schedule reliability criterion* which is to maximize reliability of the schedule. The goal is be to find such schedule that the probability of two jobs assigned to a single machine at the same time will be the smallest. To express it in terms of problem (1.3), the goal is to maximize probability:

$$\mathcal{P}\left(\xi \in \Xi : \sum_{c\in\mathcal{C}}\sum_{j\in\mathcal{J}} y_{jc}(\xi) = 0\right)$$

The problem is formulated in as follows:

$$
\begin{aligned}
\max_{x} \quad & \mathcal{P}\left(\xi \in \Xi : \sum_{j:s_j \le t < f_j(\xi)} x_{jc} \le 1, \forall c \in \mathcal{C}, \, \forall t \in \tilde{\mathcal{T}}\right) \\
\text{s.t.} \quad & \sum_{c \in \mathcal{C}} x_{jc} = 1 \qquad \forall j \in \mathcal{J}, \\
& \sum_{j:s_j \le t < f_j^0} x_{jc} \le 1 \qquad \forall c \in \mathcal{C}, \, \forall t \in \tilde{\mathcal{T}}, \\
& x_{jc} \in \{0,1\} \quad \forall c \in \mathcal{C}, \, \forall j \in \mathcal{J}
\end{aligned}
\tag{1.4}
$$

This problem can be reformulated as deterministic problem. For that we need assumption that the multivariate distribution of random delays $D_j(\xi)$ follows a copula which belongs to a special class. For definitions of copula we will use McNeil and Nešlehová [2009].

Definition 1. *A (d-dimensional) copula is a function $C : [0,1]^n \to [0,1]$ satisfying*

1. *$C(u_1, \dots, u_d) = 0$ whenever $u_i = 0$ for at least one $i \in \{1, \dots, d\}$,*
2. *$C(u_1, \dots, u_d) = u_i$ if $u_j = 0 \ \forall j \in \{1, \dots, d\}, j \neq i.$,*
3. *C is d-decreasing, i.e. for each $B = \prod_{i=1}^{J} [a_i, b_i] \subseteq [0,1]^d$*

$$\int_B dC(u) = \sum_{z \in \times_{i=1}^{d} \{a_i, b_i\}} (-1)^{card\{i : z_i = a_i\}} C(z) \ge 0.$$

According to Sklar's theorem, for every d-dimensional distribution function F there exist a copula function C for which it holds

$$F(x_1, \dots, x_d) = C\left(F_1(x_1), \dots, F_d(x_d)\right),$$

where $F_1, \dots, F_d$ are marginal distributions of F.

To represent maximal reliability FIS problem as deterministic problem we need the copula function C associated with distribution of random delays to belong to Archimedian copula class.

Definition 2. *A d-dimensional copula C belongs to the Archimedian copula class if there exists a strictly decreasing continuous function $\psi(u) : [0,1] \to \mathbb{R}^+$ satisfying:*

1. *$\psi(1) = 0$,*
2. *$\lim_{x \to 0^+} \psi(x) = \infty$,*
3. *$C(u) = \psi^{-1}\left(\sum_{i=1}^{d} \psi(u_i)\right)$*

The function $\psi(x)$ is called Archimedean generator of copula C.

Using assumption that the distribution of random delays follows Archimedean copula, we can now reformulate the maximum reliability FIS problem using graph coloring approach. Firstly we need to establish how will our graph look.

Vertices in the graph stand for jobs and colours represent machines. The edges in the graph are split into two sets. Firstly there is set E of edges which can not be colored by the same color (processing times overlap with probability 1). That means

$$\{i, j\} \in E \text{ if } s_i \leq s_j < f_i^0.$$

The second set is set $\bar{E}$ of edges which can be colored by same color (processing times overlap with probability less than 1). That means

$$\{i, j\} \in \bar{E} \text{ if } f_i^0 \leq s_j.$$

Each edge $\{i, j\} \in \bar{E}$ is associated witch a penalty $q_{ij} = \psi(1 - p_{ij})$ where p_{ij} represents probability that jobs i and j will overlap i.e. $p_{ij} = \mathcal{P}(D_i(\xi) > s_j - f_i^0)$ When we mention the pair of jobs (vertices) $\{i, j\}$ we always consider the pair to be ordered by starting times $s_i \leq s_j$.

The deterministic reformulation of the problem is then:

$$\begin{aligned}
\min_{x,y,z} \quad & \sum_{\{i,j\} \in \bar{E}} q_{ij} z_{ij} \\
\text{s.t.} \quad & \sum_{c \in \mathcal{C}} x_{jc} = 1 && \forall j \in \mathcal{J}, \\
& x_{ic} + x_{jc} \leq 1 && \forall c \in \mathcal{C},\ \forall \{i, j\} \in E, \\
& x_{ic} + x_{jc} \leq 1 + y_{ij} && \forall c \in \mathcal{C},\ \forall \{i, j\} \in \bar{E}, \\
& y_{ij} + \sum_{k:\{i,k\} \in \bar{E} \,\&\, s_k \geq f_j^0} z_{ik} \leq 1 && \forall \{i, j\} \in \bar{E}, \\
& \sum_{k:\{j,k\} \in \bar{E}} y_{jk} \leq J \sum_{k:\{j,k\} \in \bar{E}} z_{jk} && \forall j \in \mathcal{J}, \\
& x_{jc} \in \{0, 1\} && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\
& y_{ij} \in \{0, 1\} && \forall \{i, j\} \in \bar{E}, \\
& z_{ij} \in \{0, 1\} && \forall \{i, j\} \in \bar{E}
\end{aligned} \tag{1.5}$$

The first constraint ensures that each job is assigned to exactly one machine. The second constraint says that job with overlapping processing times can not be processed by the same machine. In the third constraint we introduce a new variable y_{ij} which has value $y_{ij} = 0$ if jobs i and j are scheduled to different machines and $y_{ij} = 1$ if jobs i and j are scheduled to the same machine.

In the fourth constraint

$$y_{ij} + \sum_{k:\{i,k\} \in \bar{E} \,\&\, s_k \geq f_j^0} z_{ik} \leq 1 \qquad \forall \{i, j\} \in \bar{E}$$

we introduce another new variable z_{ij} which has value $z_{ij} = 1$ if job j is scheduled at the same machine as job i and there is no other job scheduled to the same machine between these two jobs. In all other cases it is $z_{ij} = 0$. The equation says that if jobs $\{i, j\}$ are scheduled to the same machine, all jobs with starting times that are greater or equal finishing time of job j can not be immediate successors of job i.

The fifth constraint

$$\sum_{k:\{j,k\}\in\bar{E}} y_{jk} \leq J \sum_{k:\{j,k\}\in\bar{E}} z_{jk} \qquad \forall j \in \mathcal{J}$$

ensures that if there is at least one job assigned to the same machine after job j (left side of equation grater than zero) then at least one of the jobs which start after job j has to be its immediate successor.

The objective function comes from the fact that we want to maximize reliability which is probability that for each pair of consecutive jobs scheduled to the same machine ($z_{ij} = 1$) it will hold that $f_i(\xi) \leq s_j$. Using assumption that the distribution of random delays follows Archimedean copula it holds for the reliability of the schedule $R(x)$ that

$$R(x) = \psi^{-1}\left(\sum_{\{i,j\}\in\bar{E}} \psi(1-p_{ij})z_{ij}\right) = \psi^{-1}\left(\sum_{\{i,j\}\in\bar{E}} q_{ij}z_{ij}\right).$$

Since ψ is strictly monotonous decreasing function, we maximize the reliability if we minimize the objective function of (1.5)

1.2.2 Minimizing expected number of overlaps

The reliability may not be the best criterion when making repeating schedules. To maximize long term profit minimizing expected value of number of overlaps might be a better approach. *Expected number of overlaps criterion* is the second approach proposed in Branda et al. [2016]. The problem is basically problem (1.3) with addition of expected value of objective function.

$$\mathbb{E}_\xi\left[\sum_{c\in\mathcal{C}}\sum_{j\in\mathcal{J}} y_{jc}(\xi)\right]$$

Problem is that this formulation still contains stochasticity in the constraint

$$\sum_{k:f_j\leq s_k<f_j(\xi)} x_{kc} \leq y_{jc}(\xi) + J(1-x_{jc}) \qquad \forall c\in\mathcal{C},\ \forall j\in\mathcal{J},\ \forall\xi\in\Xi.$$

and in the variable $y_{jc}(\xi)$. This can be solved by for example by generating all the scenarios, which can be defined by the number of jobs that each job will overlap with due to the random delay. Problem with this approach is that there are $J!$ scenarios which is computationally impossible to solve for large problems.

It turns out that graph coloring reformulation can be used for this problem as well. The graph coloring problem is formulated

$$
\begin{aligned}
\min_{x,\,y} \quad & \sum_{\{i,j\} \in \bar{E}} q_{ij} y_{ij} & & \\
\text{s.t.} \quad & \sum_{c \in \mathcal{C}} x_{jc} = 1 & & \forall j \in \mathcal{J}, \\
& x_{ic} + x_{jc} \leq 1 & & \forall c \in \mathcal{C},\ \forall \{i,j\} \in E, \\
& x_{ic} + x_{jc} \leq 1 + y_{ij} & & \forall c \in \mathcal{C},\ \forall \{i,j\} \in \bar{E}, \\
& x_{jc} \in \{0,1\} & & \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\
& y_{ij} \in \{0,1\} & & \forall \{i,j\} \in \bar{E}
\end{aligned}
\tag{1.6}
$$

where $q_{ij} = p_{ij}$ is probability of jobs i and j to overlap.

The formulation of problem (1.6) looks very similar to problem (1.5). Even the variables x_{jc} and y_{ij} have the same meaning. The difference is that the penalty q_{ij} holds a different meaning and is activated for each pair of the jobs that are scheduled to the same machine and not only for consecutive jobs. The proof that this formulation solves the expected number of overlaps problem can be found in Branda et al. [2016] *Proposition 3.1.*

This formulation can be easily generalized to problem where different jobs are associated with different penalty if they are not performed. Let the penalties for not performing each job be $r_j,\ j \in \mathcal{J}$. Using that we modify the objective function to

$$\sum_{\{i,j\} \in \bar{E}} q_{ij} y_{ij} r_j$$

and now the problem (1.6) minimizes not expected number of overlaps between jobs but expected penalty of the schedule.

1.2.3 Network flow formulation of FIS problem

In previous parts we presented graph coloring reformulations (1.5) and (1.6) of fixed interval scheduling problem. Now we show network flow approach to this problem. The graph in this problem will look differently than in the coloring problem approach. This approach was presented in Branda and Hájek [2017]. It is superior to formulation (1.5) because due to properties of network flow problem the decision variables can be relaxed to real numbers and therefore the problem does not need to use mixed integer programming.

Now we will construct graph with $2J + 1$ vertices, which represent starting and finishing times of each job and vertices 0 and $2J + 1$ which correspond to the beginning and end of the schedule. The first set of oriented edges G contains all pairs of vertices $\{0, s_j\}$, $\{s_i, f_j\}$, $\{f_j, 2J + 1\}$ for each of the jobs $j \in \mathcal{J}$. The second set of oriented edges $\bar{G}$ contains all pairs $\{f_i, s_j\}$ where it holds that $s_j \geq f_i^0$. Lets note that s_j represents both starting time of job j and starting node of job j in graph while finishing time is represented by f_j^0 (scheduled finishing time), $f_j(\xi)$ (actual finishing time which is dependent on realization of random delay - not used in this formulation) and finishing node of job j is represented by f_j.

For each of the nodes we need to specify set of predecessors and set of successors. For each vertex u we assign set of predecessors $G_{\cdot u}$ and a set of successors $G_{u\cdot}$.

$$G_{\cdot u} = \left\{ v : \{v, u\} \in G \cup \bar{G} \right\}$$
$$G_{u\cdot} = \left\{ v : \{u, v\} \in G \cup \bar{G} \right\}$$

Note that for the 0 node the set of successors contains all the starting time nodes, for the starting time nodes the set of successors contains only one element, finishing time node of the same jobs and for finishing time nodes the set of successors contains schedule end node and starting time nodes of jobs which can be scheduled after the job to the same machine.

Next step is to assign demand d to each node. Firstly we have C machines, therefore we can have C flows from start to the machines: $d_0 = C$. Each job has to start, therefore starting time node has to accept demand from starting node 0 or finishing time node of previous job. The demand will be therefore $d_{s_j} = -1$ for all jobs $j \in \mathcal{J}$. Finishing time nodes will be associated on the other hand with demand $d_{f_j} = 1$ for all jobs $j \in \mathcal{J}$. Finally the $2J + 1$ node which represents end of the schedule has to accept each flow therefore the demand will be $d_{2J+1} = -C$.

Each of the edges $\{f_i, s_j\} \in \bar{G}$ is associated with penalty

$$q_{f_i s_j} = \psi\left(\mathcal{P}(D_i(\xi) \le s_j - f_i^0)\right)$$

which is equivalent to the penalty q_{ij} in problem (1.5).

The formulation of network-flow problem is:

$$\begin{aligned}
\min_{z} \quad & \sum_{\{f_i,s_j\}\in\bar{G}} q_{f_i s_j} z_{f_i s_j} \\
\text{s.t.} \quad & \sum_{v\in G_{0\cdot}} z_{0v} = C, \\
& \sum_{v\in G_{s_j\cdot}} z_{s_j v} - \sum_{u\in G_{\cdot s_j}} z_{u s_j} = -1 \qquad \forall j \in \mathcal{J}, \\
& \sum_{v\in G_{f_j\cdot}} z_{f_j v} - \sum_{u\in G_{\cdot f_j}} z_{u f_j} = 1 \qquad \forall j \in \mathcal{J}, \\
& \sum_{u\in G_{\cdot(2J+1)}} z_{u(2J+1)} = C, \\
& 0 \le z_{uv} \le 1 \qquad \forall \{u, v\} \in \bar{G}
\end{aligned} \tag{1.7}$$

This problem is equivalent to the formulation (1.4). The objective is to maximize reliability of the schedule.

1.3 Minimizing number of not processed jobs

In this section we would like to present an alternative criterion to the one presented in section 1.2. We will not focus on number of overlaps but on number of jobs that would not be processed. The difference is that for example let's imagine 3 jobs scheduled to single machine. It can happen that after considering random delays job one will overlap with jobs two and three and also jobs two and three will overlap. That gives us three overlaps but in reality there are only two jobs that are not completed. Therefore in this section we will not focus on the number of jobs that are overlapped by delay of each job but on the fact whether that job is processed or not.

Advantage of this approach is that number of not processed jobs can be better linked to penalty for not processing these jobs or cost of outsourcing the job than number of overlaps. The reason is that one job can be overlapped by multiple previous jobs and therefore we might penalise this job more than once.

We will need to introduce a random variable $T_{ij}(\xi) = \mathbb{I}[D_i(\xi) > s_j - f_i^0]$ which is binary random variable that indicates whether processing times of jobs i and j are overlapping. The stochastic formulation of the problem is following:

$$
\begin{aligned}
\min_{x,y,z} \quad & \sum_{j\in\mathcal{J}} z_j(\xi) \\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}} x_{jc} = 1 && \forall j \in \mathcal{J}, \\
& \sum_{j:s_j\le t<f_j^0} x_{jc} \le 1 && \forall c \in \mathcal{C}, \forall t \in \tilde{\mathcal{T}}, \\
& x_{jc} \in \{0,1\} && \forall c \in \mathcal{C}, \forall j \in \mathcal{J}, \\
& \sum_{c\in\mathcal{C}} x_{jc}x_{ic} = y_{ij} && \forall (i,j) : s_i < s_j, \\
& \sum_{i:s_i<s_j} (1 - z_i(\xi)) y_{ij} T_{ij}(\xi) = z_j(\xi) && \forall j \in \mathcal{J}
\end{aligned}
\tag{1.8}
$$

The first three constraints are standard constraints that ensure feasibility of the schedule. The forth constraint is there to introduce binary variable y_{ij} which gets activated when both jobs are scheduled to the same machine. The last constraint introduces a new binary variable $z_j(\xi)$ that indicates whether job j is done ($z_j(\xi) = 0$) or is overlapping with a previous job that is scheduled and done on the same machine. ($z_j(\xi) = 1$) after observing the random delay.

The variable $z_j(\xi)$ gets activated ($z_j(\xi) = 1$) when one of the previous jobs that is processed ($z_i(\xi) = 0$), is scheduled to the same machine ($y_{ij} = 1$) and their processing times overlap ($T_{ij}(\xi) = 1$). It can bee easily verified that $z_j(\xi)$ can not be grater than 1. Imagine that $z_j(\xi) \ge 2$ that means there are jobs i and k for which all of the above holds. Let's say $s_i < s_k < s_j$. We have $y_{ij} = y_{kj} = 1$ therefore also $y_{ik} = 1$ because pair of jobs i and k is also scheduled to the same machine. We have $T_{ij}(\xi) = 1$ but since $s_k < s_j$ also $T_{ik}(\xi) = 1$. Therefore we will get that $z_k(\xi) = 1$ which means that $z_j(\xi)$ can not be activated by both jobs i and k.

1.3.1 Expected number of not processed jobs

The easiest way how to deal with stochasticity in problem (1.8) is to switch to expected numbers. For long term repeating schedules is expected value suitable measure as it minimizes overall long term costs (penalties).

In this formulation we will use the same graph coloring notation as used in section (1.2.1) where we have jobs as vertices and machines as color. The edges are split into two sets E and $\bar{E}$ where first set E contains pairs of jobs which can not be scheduled to the same machine and the second set $\bar{E}$ contains pair of jobs that do not overlap with positive probability. We will also use assumption that delays of the jobs are independent.

The linear programming reformulation for expected value objective is following:

$$
\begin{array}{lrll}
\min\limits_{x,y,z} & \sum\limits_{j\in\mathcal{J}} z_j & & \\
\text{s.t.} & \sum\limits_{c\in\mathcal{C}} x_{jc} = 1 & \forall j \in \mathcal{J}, & \\
& x_{jc} \in \{0,1\} & \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, & \\
& x_{ic} + x_{jc} \leq 1 & \forall c \in \mathcal{C},\ \forall \{i,j\} \in E, & \\
& x_{ic} + x_{jc} \leq 1 + y_{ij} & \forall c \in \mathcal{C},\ \forall \{i,j\} \in \bar{E}, & \\
& y_{ij} + y_{ik} \leq 1 + y_{jk} & \forall i,j,k \in \mathcal{J}, & \\
& y_{ij} \in \{0,1\} & \forall \{i,j\} \in \bar{E}, & (1.9) \\
& y_{ij} = 0 & \forall \{i,j\} \in E, & \\
& z_j = \sum\limits_{i:\{i,j\}\in\bar{E}} z^*_{ij} & \forall j \in \mathcal{J}, & \\
& z^*_{ij} \geq 0 & \forall \{i,j\} \in \bar{E}, & \\
& 1 + z^*_{ij} \geq (1 - z_i)p_{ij} + y_{ij} & \forall \{i,j\} \in \bar{E}, & \\
& z^*_{ij} \leq y_{ij} & \forall \{i,j\} \in \bar{E}, & \\
& z^*_{ij} \leq (1 - z_i)p_{ij} & \forall \{i,j\} \in \bar{E} &
\end{array}
$$

The first set of constraints (for variables x) is universal for all FIS problems, it constraints us to the set of feasible schedules. The second set of constraints (for variable y) is there to determine variable y_{ij} as binary indicator that jobs i and j are scheduled to the same machine.

First of the constraints $x_{ic} + x_{jc} \leq 1 + y_{ij}$ forces $y_{ij} = 1$ if they are scheduled to the same machine. Second one is there to ensure that if jobs are scheduled to different machines then $y_{ij} = 0$. If $y_{ij} = 1$ where job i is scheduled to machine 1 and job j to machine 2 and let job k be scheduled to machine 2 then $y_{jk} = 1$ because they are both scheduled to the same machine but because of the constraint $y_{ij} + y_{jk} \leq 1 + y_{ik}$ also $y_{ik} = 1$ even tho they are scheduled to different machines. Therefore if $y_{ij} = 1$ then for all pairs jobs where job k is scheduled to machine 2 and job l is scheduled to machine 1 it will hold that $y_{kl} = 1$. If one of the pairs of jobs overlaps with probability 1 then such solution would be infeasible because of last constraint of the set. ($y_{ij} = 0$ if jobs i and j overlap with probability 1.) In case none of the jobs overlap it would create penalty equivalent to schedule where all jobs from these machines are scheduled to machine 1 and nothing is scheduled to machine 2 and it can be easily seen that such schedule can not have smaller penalty.

This set of extra constraints for variables y was not necessary in previous formulations, the difference is that in previous formulations the variables y_{ij} were connected only to positive values in objective function and therefore it was set to 0 by minimization in case the jobs were not scheduled to the same machine. In this formulation the variables are connecting to each other in more complicated ways and therefore we need to ensure that $y_{ij} = 0$ in case the jobs are scheduled to different machines.

Last set of constraints introduces our penalties, which are probabilities that job j is overlapped by one of the previous jobs conducted on the same machine. As it was explained in the previous section job can be not conducted only because of one of the previous jobs scheduled to the same machine. Therefore using this and independence of the delays we get that

$$\begin{aligned}
z_j = \mathcal{P}(z_j(\xi) = 1) = \mathcal{P}\left(\sum_{i:s_i<s_j} (1 - z_i(\xi))y_{ij}T_{ij}(\xi) = 1\right) && (1.10)\\
= \sum_{i:s_i<s_j} y_{ij}\mathcal{P}\left((1 - z_i(\xi))T_{ij}(\xi) = 1\right) && (1.11)\\
= \sum_{i:s_i<s_j} y_{ij}\mathcal{P}\left((1 - z_i(\xi)) = 1\right)\mathcal{P}\left(T_{ij}(\xi) = 1\right) && (1.12)\\
= \sum_{i:s_i<s_j} y_{ij}(1 - z_i)p_{ij} && (1.13)\\
= \sum_{i:\{i,j\}\in\bar{E}} z^*_{ij} && (1.14)
\end{aligned}$$

The last equation comes from the last four constraints. If $y_{ij} = 0$ then also $z^*_{ij} = 0$. If $y_{ij} = 1$ then $z^*_{ij} = (1 - z_i)p_{ij}$. Variable z^*_{ij} is therefore probability that job i causes job j to be unprocessed.

What needs to be proven is that $z_j = \mathcal{P}(z_j(\xi) = 1)$. For the first job it holds that it is processed certainly because there can not be any job scheduled before job 1. Therefore $\mathcal{P}(z_1(\xi) = 1) = 0$. For z_1 it holds that $z_1 = 0$ because again no job can be scheduled before job 1 and therefore $\{i : \{i, j\} \in \bar{E}\}$ is empty set. That means the equality holds for $j = 1$. In equation (1.10) we proved that if it holds for all $i < j$ then it holds also for j. Since we also proved that it holds for $j = 1$ from mathematical induction it holds for all j.

To generalize this problem to situation where the jobs are associated with different penalties when they are unprocessed we can simply modify the objective function by adding penalties. Lets say that penalty for not doing job j is $r_j > 0$, then we can simply change the objective function to

$$\min_{x,y,z} \quad \sum_{j\in\mathcal{J}} z_j r_j$$

In this modified problem we will be looking for schedule with the lowest expected penalty.

2. Risk measures

The problem we are dealing in this thesis is stochastic in its nature. The outcome of the schedule, actual number of jobs that will not be executed is random variable which is dependent on random delays. In such problems it can be useful to introduce risk measures into the problem. While maximizing expected value is usually reasonable approach when we want to create multiple repeating strategies, it comes from the law of big numbers, it can be also associated with possible big losses in some of the schedules. Of course in the long run the higher expected value compensates the possible losses but in the short term the losses can be big enough to cross some acceptable threshold.

Now we need too introduce what a risk measure is. We will use definitions from Shapiro et al. [2009].

Definition 3. *Risk measure. Let $Z(\xi)$ be a random variable from probability space $(\Xi, \mathcal{A}, \mathcal{P})$. By a risk measure we understand function $\rho(Z)$ which assigns the random variable Z a value from $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty, -\infty\}$.*

The definition states that risk measure can be any function from space of random variables to $\bar{\mathbb{R}}$. One of the basic ones can be variance. Variance as a risk measure is used for example in *Markowitz portfolio* problem, Markowitz [1952], which is a bi-criteria optimization of revenue $R(x)$ over set of possible portfolios $\mathcal{X}$.

$$\max_x \quad \mathbb{E}[R(x)]$$

$$\begin{aligned} \min_x \quad & var(R(x)) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned}$$

the solution to this problem is a set of efficient portfolios (efficient frontier) and the problem can be solved by help of parametric optimization. Now we present parametric formulation for bi-criteria mean-risk problem. Unlike in Markowitz portfolio problem we will consider the random variable $Z(\xi, x)$ to represent loss or something else we would like to minimize. The first formulation uses parameter only in objective function.

$$\begin{aligned} \min_x \quad & \mathbb{E}_\xi[Z(\xi, x)] + \lambda \rho_\xi(Z(\xi, x)) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned} \tag{2.1}$$

where $\lambda > 0$ is a parameter and $\mathcal{X}$ is the set of feasible solutions.

The second and third formulations put the parameter into Constraints as maximal acceptable expected value Z_{max}

$$\begin{aligned} \min_x \quad & \rho_\xi(Z(\xi, x)) \\ \text{s.t.} \quad & \mathbb{E}_\xi[Z(\xi, x)] \leq Z_{max}, \\ & x \in \mathcal{X} \end{aligned} \tag{2.2}$$

or maximal accepted risk r_{max}.

$$\begin{aligned} \min_x \quad & \mathbb{E}_\xi[Z(\xi,x)] \\ \text{s.t.} \quad & \rho_\xi(Z(\xi,x)) \le r_{max}, \\ & x \in \mathcal{X} \end{aligned} \tag{2.3}$$

From the multi criterion optimization theory we know that the solution to problem (2.1) where parameter $\lambda > 0$ is always efficient. For problems (2.2) and (2.3) we can not say the same as there may not be unique solution to the problem and not all solutions have to be efficient. Therefore for our mean-risk analysis we will be using first formulation.

2.0.1 Coherent risk measures

In order to use the risk measure for some analysis we would like it to fulfill some conditions. That brings us to introduction of a subset of the risk measures called coherent risk measures. Firstly we need to define inequality between random loss variables:

$$Z \succeq Z' \iff \forall \xi \in \Xi : Z(\xi) \ge Z'(\xi)$$

Now we are ready to define the coherent risk measures. In Shapiro et al. [2009] the definition of coherent risk measures is following.

Definition 4. *Coherent risk measure*

Let Z, Z' be random variables from probability space $(\Xi, \mathcal{A}, \mathcal{P})$ and $\rho(Z)$ be a risk measure. We say that $\rho(Z)$ is coherent risk measure if following conditions hold $\forall Z$, $Z' \in (\Xi, \mathcal{A}, \mathcal{P})$:

1. *Convexity:*

$$\rho(tZ + (1-t)Z') \le t\rho(Z) + (1-t)\rho(Z') \qquad \forall t \in [0,1].$$

2. *Monotonicity:*

$$Z \succeq Z' \implies \rho(Z) \ge \rho(Z').$$

3. *Translation equivalence:*

$$\rho(Z + a) = \rho(Z) + a \qquad \forall a \in \mathbb{R}.$$

4. *Positive homogenity:*

$$\rho(tZ) = t\rho(Z) \qquad \forall t > 0.$$

The most common coherent risk measure is conditional value at risk (*CVaR*). On the other hand popular risk measures like variance or value at risk are not coherent.

2.1 Commonly used risk measures

In this section we will present some risk measures that are commonly used.

2.1.1 Variance

One of the most basic risk measured is variance. The reasoning why it is useful comes from the application in the mean-risk models. Lowering variance means that the random outcome will be closer to the mean and therefore we will get more stable solution. On the other hand, variance does not fulfill all the conditions of coherent risk measure.

Theorem 1. *Variance as risk measure fulfills only condition of convexity. All the other conditions: monotonicity, translation equivalence and positive homogenity are not fulfilled. Therefore variance is not a coherent risk measure.*

Proof. Let Z, Z' be random variables from probability space $(\Xi, \mathcal{A}, \mathcal{P})$.

1. Convexity: let $t \in [0,1]$

$$\begin{aligned}
&var(tZ + (1-t)Z') = t^2 var(Z) + (1-t)^2 var(Z') + 2t(1-t)cov(Z,Z') \\
&\leq t^2 var(Z) + (1-t)^2 var(Z') + 2t(1-t)sd(Z)sd(Z') \\
&= t\,var(Z) + (1-t)var(Z') + (t^2 - t)var(Z) + \\
&\qquad\qquad + ((1-t)^2 - (1-t))var(Z') + 2t(1-t)sd(Z)sd(Z') \\
&= t\,var(Z) + (1-t)var(Z') - t(1-t)var(Z) - \\
&\qquad\qquad - t(1-t)var(Z') + 2t(1-t)sd(Z)sd(Z') \\
&= t\,var(Z) + (1-t)var(Z') - t(1-t)[sd(Z)^2 + sd(Z')^2 - 2sd(Z)sd(Z')] \\
&= t\,var(Z) + (1-t)var(Z') - t(1-t)[sd(Z) - sd(Z')]^2 \\
&\leq t\,var(Z) + (1-t)var(Z').
\end{aligned}$$

   Therefore variance is convex.

2. Monotonicity:

   Let $Z \sim Alt(0,1)$ with probability $\mathcal{P}(Z=1) = 0.5$ and $Z' \sim Alt(2,3)$ with probability $\mathcal{P}(Z'=3) = 0.1$. It is obvious that $Z' \succeq Z$ but $var(Z) = 0.25 > 0.09 = var(Z')$.

   Therefore variance is not monotonous.

3. Translation equivalence: let $a \neq 0$

$$var(Z+a) = var(Z) \neq var(Z) + a.$$

   Therefore variance is not translation equivalent.

4. Positive homogenity: let $t \neq 1$

$$var(tZ) = t^2 var(Z) \neq t\,var(Z).$$

   Therefore variance is not positively homogeneous.

Because not all condition are fulfilled, variance is not a coherent risk measure.

□

2.1.2 Standard deviation

Another of the basic risk measures is standard deviation. It is used mostly in finance where they called it volatility. Since the standard deviation is just square root of variance transformation between those two measures is monotonous. As a result of this minimizing variance and minimizing standard deviation is the same. Also the efficient frontier for mean-variance and mean-standard deviation bi-variate problem is the same. Those two risk measures can be used interchangeably. Standard deviation is better risk measure than variance because compared to variance it is also positive homogeneous.

Theorem 2. *Standard deviation as risk measure is convex and positively homogeneous. It does not fulfill conditions of monotonicity and translation equivalence therefore it is not a coherent risk measure.*

Proof. Let Z, Z' be random variables from probability space $(\Xi, \mathcal{A}, \mathcal{P})$.

1. Convexity: let $t \in [0, 1]$

$$\begin{aligned}&[sd(tZ + (1-t)Z')]^2 = var(tZ + (1-t)Z')\\ &= t^2 var(Z) + (1-t)^2 var(Z') + 2t(1-t)cov(Z, Z')\\ &\leq t^2[sd(Z)]^2 + (1-t)^2[sd(Z')]^2 + 2t(1-t)sd(Z)sd(Z')\\ &= [t\, sd(Z) + (1-t)sd(Z')]^2\end{aligned}$$

   Applying square root on both sides we get that standard deviation is convex.

2. Monotonicity:

   Using same example as with variance, let $Z \sim Alt(0, 1)$ with probability $\mathcal{P}(Z = 1) = 0.5$ and $Z' \sim Alt(2, 3)$ with probability $\mathcal{P}(Z' = 3) = 0.1$. It is obvious that $Z' \succeq Z$ but $sd(Z) = 0.5 > 0.3 = sd(Z')$.

   Therefore standard deviation is not monotonous.

3. Translation equivalence: let $a \neq 0$

$$sd(Z + a) = sd(Z) \neq sd(Z) + a.$$

   Therefore variance is not translation equivalent.

4. Positive homogenity: let $t > 0$

$$sd(tZ) = t\, sd(Z).$$

   Standard deviation is positively homogeneous.

Because not all condition are fulfilled, standard deviation is also not a coherent risk measure.

□

However standard deviation is not a coherent risk measure, it fulfills all the axioms of a *deviation measure* defined in Rockafellar et al. [2006].

2.1.3 Conditional Value at Risk

The most important risk measure we will focus on in this thesis is *Conditional Value at Risk* in some literature also denoted as *Expected shortfall.* $CVaR_\alpha$ where α is called confidence level can be understood as expected value in worst-case situations that occur with probability $1-\alpha$. Common choice for confidence level is $\alpha = 0.95$ therefore $CVaR_{0.95}$ is expected value of 5% highest losses. In Rockafellar and Uryasev [2002] it is proven that $CVaR_\alpha$ is a coherent risk measure.

In this thesis we want to optimize number of overlaps and number of unfulfilled jobs. Those are both discrete random variables therefore we can focus more on *CVaR* for discrete random variables.

Let Z be a discrete random variable with finite set of random values: $z_1 < z_2 < \cdots < z_N$ with probabilities $p_1, \ldots, p_N$. Now let n_0 be such that

$$\sum_{i=1}^{n_0-1} p_i < \alpha \leq \sum_{i=1}^{n_0} p_i$$

then it holds that

$$CVaR_\alpha(Z) = \frac{1}{1-\alpha}\left[\left(\sum_{i=1}^{n_0} p_i - \alpha\right) z_{n_0} + \sum_{i=n_0+1}^{N} p_i z_i\right] \tag{2.4}$$

Better way to calculate *CVaR* is to use minimization formula. If Z is a random variable with finite expected value, using minimization formula, the $CVaR_\alpha$ can be calculated as

$$CVaR_\alpha(Z) = \min_a \; a + \frac{1}{1-\alpha}\mathbb{E}[Z-a]^+$$

where $[Z-a]^+ = max(0, Z-a)$

3. Risk measures in FIS problem

In this chapter we will combine the Fixed interval scheduling from chapter 1 with risk measures presented in chapter 2. The aim is to find new ways how to deal with fixed interval scheduling while taking into account uncertainty.

3.1 Mean-variance optimization of number of overlaps

The first formulation we will focus on is mean-variance optimization. The problem is similar to *Markowitz portfolio* problem. Instead of returns from investment we will look at number of overlaps between jobs scheduled to the same machine, which will pay the role of random losses. In problem (1.6) we minimized expected value of the overlaps, now we would like to minimize the expected number as well as the variance of the number of overlaps. Since it is bi-criteria optimization, we will use formulation (2.1) with parameter $\lambda > 0$.

In context of problem (1.3) we would like to minimize following objective function:

$$\mathbb{E}_\xi \left[\sum_{c\in\mathcal{C}} \sum_{j\in\mathcal{J}} y_{jc}(\xi) \right] + \lambda \, var \left(\sum_{c\in\mathcal{C}} \sum_{j\in\mathcal{J}} y_{jc}(\xi) \right), \qquad \lambda > 0 \tag{3.1}$$

The notation is based on the same graph coloring problem as in (1.6) and is expanded with new notation specific for this problem. The first formulation of this problem assumes that the random delays of the jobs are pairwise uncorrelated.

$$\begin{aligned} \min_{x,\,y,\,z} \quad & \sum_{\{i,j\}\in\bar{E}} y_{ij}(q_{ij} + \lambda v_{ij}) + 2\lambda \sum_{(\{i,j\}\&\{j,k\})\in\bar{E}} z_{ijk} v_{ijk} \\ \text{s.t.} \quad & \sum_{c\in\mathcal{C}} x_{jc} = 1 && \forall j \in \mathcal{J}, \\ & x_{ic} + x_{jc} \le 1 && \forall c \in \mathcal{C},\ \forall \{i,j\} \in E, \\ & x_{ic} + x_{jc} \le 1 + y_{ij} && \forall c \in \mathcal{C},\ \forall \{i,j\} \in \bar{E}, \\ & y_{ij} + y_{ik} \le 1 + z_{ijk} && \forall (\{i,j\}\&\{j,k\}) \in \bar{E}, \\ & x_{jc} \in \{0,1\} && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\ & y_{ij} \in \{0,1\} && \forall \{i,j\} \in \bar{E}, \\ & z_{ijk} \in \{0,1\} && \forall (\{i,j\}\&\{j,k\}) \in \bar{E} \end{aligned} \tag{3.2}$$

where $q_{ij} = p_{ij}$ expresses probability of jobs i and j overlapping (expected value of that event), $v_{ij} = p_{ij}(1-p_{ij})$ is variance of event that these jobs would overlap and $v_{ijk} = p_{ik}(1-p_{ij})$ is covariance between event that jobs i and j are overlapping and jobs i and k are overlapping. Additional variables introduced in this formulation z_{ijk} are indicators that the triplet of jobs i, j and k are all scheduled to the same machine.

Theorem 3. *Problem (3.2) minimizes the objective function (3.1) over set of feasible schedules. The set of feasible schedules is equivalent to set of feasible solutions from problem (1.3).*

Proof. It is easy to note that the constrains of both problems define the same set of feasible schedules. The only thing we need to do therefore is to compare the objective functions.

From Branda et al. [2016] *Proposition 3.1* we already know that the first part of the objective function (3.1) corresponds to

$$\mathbb{E}_\xi \left[\sum_{c \in \mathcal{C}} \sum_{j \in \mathcal{J}} y_{jc}(\xi) \right] = \sum_{\{i,j\} \in \bar{E}} y_{ij} q_{ij}$$

Now we will look at the second part.

$$\begin{aligned} &var\left(\sum_{c\in\mathcal{C}}\sum_{j\in\mathcal{J}} y_{jc}(\xi)\right) = \\ &= \sum_{j\in\mathcal{J}} var\left(\sum_{c\in\mathcal{C}} y_{jc}(\xi)\right) + 2\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{J}, i\neq j} cov\left(\sum_{c\in\mathcal{C}} y_{jc}(\xi), \sum_{c\in\mathcal{C}} y_{ic}(\xi)\right) \\ &= \sum_{j\in\mathcal{J}} var\left(\sum_{c\in\mathcal{C}} y_{jc}(\xi)\right) \end{aligned} \tag{3.3}$$

Last equivalence comes from the assumption that the delays are pairwise uncorrelated. Now we will rewrite variable $y_{ic}(\xi)$:

$$\sum_{c\in\mathcal{C}} y_{ic}(\xi) = \sum_{j:\{i,j\}\in\bar{E}} y_{ij}\mathbb{I}[D_i(\xi) > s_j - f_i^0] = \sum_{j:\{i,j\}\in\bar{E}} y_{ij} T_{ij}(\xi)$$

where y_{ij} is indicator variable that jobs i and j are scheduled to the same machine and $T_{ij}(\xi) = \mathbb{I}[D_i(\xi) > s_j - f_i^0]$ is binary random variable indicating whether their processing times are overlapping.

$$\begin{aligned} &var\left(\sum_{c\in\mathcal{C}} y_{ic}(\xi)\right) = var\left(\sum_{j:\{i,j\}\in\bar{E}} y_{ij}T_{ij}(\xi)\right) \\ &= \sum_{j:\{i,j\}\in\bar{E}} var\left(y_{ij}T_{ij}(\xi)\right) + 2\sum_{j:\{i,j\}\in\bar{E}}\sum_{k:\{j,k\}\in\bar{E}} cov\left(y_{ij}T_{ij}(\xi), y_{ik}T_{ik}(\xi)\right) \\ &= \sum_{j:\{i,j\}\in\bar{E}} y_{ij}\, var\left(T_{ij}(\xi)\right) + 2\sum_{j:\{i,j\}\in\bar{E}}\sum_{k:\{j,k\}\in\bar{E}} y_{ij}y_{ik}\, cov\left(T_{ij}(\xi), T_{ik}(\xi)\right) \end{aligned} \tag{3.4}$$

and there we can also use:

$$\begin{aligned} v_{ij} := \quad & var\left(T_{ij}(\xi)\right) = p_{ij}(1-p_{ij}) \\ v_{ijk} := \quad & cov\left(T_{ij}(\xi), T_{ik}(\xi)\right) = \mathbb{E}_\xi[T_{ij}(\xi)T_{ik}(\xi)] - \mathbb{E}_\xi[T_{ij}(\xi)]\mathbb{E}_\xi[T_{ik}(\xi)] \\ &= \mathbb{E}_\xi[T_{ik}(\xi)] - \mathbb{E}_\xi[T_{ij}(\xi)]\mathbb{E}_\xi[T_{ik}(\xi)] \\ &= p_{ik} - p_{ij}p_{ik} = p_{ik}(1-p_{ij}) \\ & y_{ij}y_{ik} = z_{ijk} \end{aligned} \tag{3.5}$$

The last equivalence hold from the fact that if both $y_{ij} = 1$ and $y_{jk} = 1$ then from the fourth constraint also $z_{ijk} = 1$ and if at least one of them is not 1 then $z_{ijk} = 0$ because of the minimization and the fact that v_{ijk} is non-negative.

Equivalence $\mathbb{E}_\xi[T_{ij}(\xi)T_{ik}(\xi)] = \mathbb{E}_\xi[T_{ik}(\xi)]$ comes from the fact that $T_{ik}(\xi)$ is a subevent of $T_{ij}(\xi)$ when $j < k$. Therefore expected value (probability) of both events happening is equal to the expected value (probability) of the subevent happening.

Combining all off this we will get that

$$\begin{aligned}
&\mathbb{E}_\xi\left[\sum_{c\in\mathcal{C}}\sum_{j\in\mathcal{J}} y_{jc}(\xi)\right] + \lambda\, var\left(\sum_{c\in\mathcal{C}}\sum_{j\in\mathcal{J}} y_{jc}(\xi)\right) = \\
&= \sum_{\{i,j\}\in\bar{E}} y_{ij}q_{ij} + \lambda \sum_{j\in\mathcal{J}} var\left(\sum_{c\in\mathcal{C}} y_{jc}(\xi)\right) \\
&= \sum_{\{i,j\}\in\bar{E}} y_{ij}q_{ij} + \lambda \sum_{i\in\mathcal{J}}\left(\sum_{j:\{i,j\}\in\bar{E}} y_{ij}\, v_{ij} + 2\sum_{j:\{i,j\}\in\bar{E}}\ \sum_{k:\{j,k\}\in\bar{E}} z_{ijk}\, v_{ijk}\right) \\
&= \sum_{\{i,j\}\in\bar{E}} y_{ij}(q_{ij} + \lambda v_{ij}) + 2\lambda \sum_{(\{i,j\}\&\{j,k\})\in\bar{E}} z_{ijk}v_{ijk}
\end{aligned}$$

Where first equation comes from previous formulation and (3.3). The second equation comes from (3.4) and (3.5). The last equation is only reformulation.

□

In conclusion, under the assumptions that the random delays of the jobs are pairwise uncorrelated, the mean-variance problem can be formulated as linear integer programming.

3.1.1 Mean-variance optimization with correlated delays

In this section we will generalize the formulation to situation where the delays can be correlated with non-negative correlation. We will add new decision variables z^*_{ijkl} which are binary indicators that pair jobs i and j is scheduled to the same machine and also pair of jobs k and l is scheduled to the same machine (both pairs can be scheduled to different machine).

$$
\begin{aligned}
\min_{x,\,y,\,z} \quad & 2\lambda \left(\sum_{(\{i,j\}\&\{j,k\})\in\bar{E}} z_{ijk} v_{ijk} + \sum_{(\{i,j\}\&\{k,l\})\in\bar{E},\, i\neq k} z^*_{ijkl} v_{ijkl} \right) \\
& + \sum_{\{i,j\}\in\bar{E}} y_{ij}(q_{ij} + \lambda v_{ij}) \\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}} x_{jc} = 1 && \forall j \in \mathcal{J}, \\
& x_{ic} + x_{jc} \leq 1 && \forall c \in \mathcal{C},\ \forall\{i,j\} \in E, \\
& x_{ic} + x_{jc} \leq 1 + y_{ij} && \forall c \in \mathcal{C},\ \forall\{i,j\} \in \bar{E}, \\
& y_{ij} + y_{ik} \leq 1 + z_{ijk} && \forall(\{i,j\}\&\{j,k\}) \in \bar{E}, \\
& y_{ij} + y_{kl} \leq 1 + z^*_{ijkl} && \forall(\{i,j\}\&\{k,l\}) \in \bar{E},\ i \neq k, \\
& x_{jc} \in \{0,1\} && \forall c \in \mathcal{C},\ \forall j \in \mathcal{J}, \\
& y_{ij} \in \{0,1\} && \forall \{i,j\} \in \bar{E}, \\
& z_{ijk} \in \{0,1\} && \forall(\{i,j\}\&\{j,k\}) \in \bar{E}, \\
& z^*_{ijkl} \in \{0,1\} && \forall(\{i,j\}\&\{k,l\}) \in \bar{E},\ i \neq k
\end{aligned}
\tag{3.6}
$$

where all the notation is the same as it was in problem (3.2) with additional parameters $v_{ijkl} = cov(T_{ij}(\xi), T_{kl}(\xi))$. The non-negativity of correlation is there to ensure that $v_{ijkl} \geq 0$.

The difference between this problem and the previous problem (3.2) is in the correlation between delays of different jobs. Therefore we need to add to the objective function

$$
2\sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{J}, i\neq j} cov\left(\sum_{c\in\mathcal{C}} y_{jc}(\xi), \sum_{c\in\mathcal{C}} y_{ic}(\xi) \right)
$$

That can be rewritten as:

$$
\begin{aligned}
cov\left(\sum_{c\in\mathcal{C}} y_{kc}(\xi), \sum_{c\in\mathcal{C}} y_{kc}(\xi) \right) &= cov\left(\sum_{j:\{i,j\}\in\bar{E}} y_{ij} T_{ij}(\xi), \sum_{k:\{k,l\}\in\bar{E}} y_{kl} T_{kl}(\xi) \right) \\
= \sum_{j:\{i,j\}\in\bar{E}} \sum_{k:\{k,l\}\in\bar{E}} y_{ij} y_{kl}\, cov\left(T_{ij}(\xi), T_{kl}(\xi)\right) \\
= \sum_{j:\{i,j\}\in\bar{E}} \sum_{k:\{k,l\}\in\bar{E}} z^*_{ijkl} v_{ijkl}
\end{aligned}
$$

last equation holds because if both $y_{ij} = 1$ and $y_{kl} = 1$ then from fifth constraint also $z^*_{ijkl} = 1$ and if one of them is not equal to 1 than from non-negativity of v_{ijkl} minimizing of objective function would push z^*_{ijkl} to 0. Alternatively $v_{ijkl} = 0$ and in that case the value of z^*_{ijkl} is not important.

3.2 CVaR of number of overlaps

In this chapter, our aim is to optimize conditional value at risk of number of overlaps. We will present a problem formulation where the optimal schedule will be the one with the lowest *CVaR*. Since the stochasticity in our problem can be decomposed into finite number of scenarios we will use it and define a set of scenarios $\mathcal{S}$. In each of the scenarios $\pi \in \mathcal{S}$ it is defined for each job $j \in \mathcal{J}$ which of the following jobs is first not to overlap with job j. We understand the set of jobs as ordered set $\mathcal{J} = \{1, \ldots, J\}$ where $i > j$ means that $s_i \geq s_j$. In case the starting times are equal we can order the jobs by scheduled finishing times or randomly. Jobs that have the same starting time will overlap with probability 1 that means none of them can be scheduled to the same machine after the other one and therefore it is not important in which order they are in set $\mathcal{J}$.

$$\pi = \{\{1, j_{1\pi}\}, \{2, j_{2\pi}\}, \ldots, \{J-1, j_{(J-1)\pi}\}, \{J, \infty\}\}$$

The pair $\{i, j_{i\pi}\}$ means that in scenario π it holds that

$$s_{j_{i\pi}-1} < f_i^0 + D_i(\xi) \leq s_{j_{i\pi}} \tag{3.7}$$

$j_{i\pi}$ can attain values $i+1, \ldots, J, \infty$ where in case $j_{i\pi} = \infty$ we understand that all following jobs are overlapping with job i. If job i is the last job $(j = J)$ then there are no following jobs that can go after it and therefore $j_{J\pi} = \infty$ in all scenarios. We can set $s_\infty = +\infty$ and the last inequality will hold also for these cases. If $j_{i\pi} = \infty$ then we will need to set $j_{i\pi} - 1 = J$ in that case as the job J is the last one to overlap with job i. For the last job J there is no successor available therefore there is only one scenario $\{J, \infty\}$.

Now let us look at the scenarios for situation where we have $J = 3$ jobs. We have 6 scenarios:

- $\{\{1,2\},\{2,3\},\{3,\infty\}\}$ - no jobs are overlapping.
- $\{\{1,2\},\{2,\infty\},\{3,\infty\}\}$ - jobs 2 and 3 are overlapping.
- $\{\{1,3\},\{2,3\},\{3,\infty\}\}$ - jobs 1 and 2 are overlapping.
- $\{\{1,3\},\{2,\infty\},\{3,\infty\}\}$ - job 1 is overlapping with job 2 and job 2 is overlapping with job 3.
- $\{\{1,\infty\},\{2,3\},\{3,\infty\}\}$ - job 1 is overlapping with jobs 2 and 3.
- $\{\{1,\infty\},\{2,\infty\},\{3,\infty\}\}$ - job 1 in overlapping with jobs 2 and 3 and also job 2 is overlapping with job 3.

For each of the scenarios we will need to compute probability of its happening. Firstly we will define $p_{i(j_{i\pi}-1)j_{i\pi}} = p_{i(j_{i\pi}-1)} - p_{ij_{i\pi}}$ is probability that 3.7 holds. The probability of each scenarios can be then computed as

$$p_\pi = \prod_{i=1}^{J} p_{i(j_{i\pi}-1)j_{i\pi}} \tag{3.8}$$

if we assume independence between random delays.

Some of the scenarios can have probability $p_\pi = 0$. For example if jobs 1 and 2 overlap with probability $p_{12} = 1$ then the first 3 scenarios from the example are impossible. Probability of the first scenario will be computed as

$$\begin{aligned} p_\pi &= p_{112}p_{223}p_{33\infty} = (p_{11} - p_{12})(p_{22} - p_{23})(p_{33} - p_{3\infty}) \\ &= (1-1)(1-p_{23})(1-0) = 0 \end{aligned}$$

To reduce number of scenarios it is possible to omit these scenarios with probability $p_\pi = 0$.

For the formulation of the problem we will use again graph coloring formulation that was first mentioned in problem (1.5). The *CVaR* problem will be formulated as follows:

$$\begin{array}{lrll} \min\limits_{x,y,a,d} & a + \frac{1}{1-\alpha}\sum\limits_{\pi\in\mathcal{S}} p_\pi d_\pi & & \\ \text{s.t.} & \sum\limits_{c\in\mathcal{C}} x_{jc} = 1 & \forall j \in \mathcal{J}, & \\ & x_{ic} + x_{jc} \le 1 & \forall c \in \mathcal{C},\ \forall\{i,j\} \in E, & \\ & x_{ic} + x_{jc} \le 1 + y_{ij} & \forall c \in \mathcal{C},\ \forall\{i,j\} \in \bar{E}, & \\ & x_{jc} \in \{0,1\} & \forall j \in \mathcal{J},\ \forall c \in \mathcal{C}, & (3.9) \\ & y_{ij} \in \{0,1\} & \forall\{i,j\} \in \bar{E}, & \\ & y_{ij} = 0 & \forall\{i,j\} \in E, & \\ & \sum\limits_{i\in\mathcal{J}}\sum\limits_{k=i+1}^{j_{i\pi}-1} y_{ik} - a \le d_\pi & \forall \pi \in \mathcal{S}, & \\ & 0 \le d_\pi & \forall \pi \in \mathcal{S} & \end{array}$$

In comparison to the previous problem (1.5) we introduced new scenario dependent variables d_π which serve as $max(0, \#overlaps - a)$. Another new real decision variable is a which comes from the optimization formulation of $CVaR_\alpha$.

Theorem 4. *The linear programming problem (3.9) will return feasible schedule with the lowest value of $CVaR_\alpha$.*

Proof. Set of feasible schedules is the same as in all previous formulations, therefore we only need to check if the objective function minimizes $CVaR_\alpha$ correctly. $CVaR_\alpha$ can be rewritten as minimization problem

$$CVaR_\alpha(Z) = \min_a\ a + \frac{1}{1-\alpha}\mathbb{E}[Z-a]^+$$

which can be combined into objective function as

$$\begin{aligned} \min_x\ CVaR_\alpha(Z) &= \min_x\ \min_a\ a + \frac{1}{1-\alpha}\mathbb{E}[Z-a]^+ \\ &= \min_{x,a}\ a + \frac{1}{1-\alpha}\mathbb{E}d \end{aligned}$$

where $d := [Z-a]^+$. Since there is finite number of scenarios $\mathbb{E}d = \sum_{\pi\in\mathcal{S}} p_\pi d_\pi$. Now it is sufficient to show that d_π is really maximum of 0 and number of overlaps under scenario π minus a. From the last constraint we know that $d_\pi \geq 0$ and from the previous one we know that it is greater or equal number of overlaps in the schedule under scenario π minus a. Since d_π is connected only to non-negative values in objective function, the value of d_π will be the lowest possible, therefore value of d_π is really what we need and the objective function minimizes $CVaR_\alpha$ of number of overlaps.

□

3.2.1 Mean-CVaR optimization of number of overlaps

Combining problem (3.9) where we minimize $CVaR_\alpha$ with problem (1.6) where we minimize expected value we will get *mean-CVaR* formulation of the problem.

$$\begin{array}{lrll}
\min\limits_{x,y,a,d} & \multicolumn{3}{l}{\sum\limits_{\{i,j\}\in\bar{E}} p_{ij}y_{ij} + \lambda\left(a + \frac{1}{1-\alpha}\sum\limits_{\pi\in\mathcal{S}} p_\pi d_\pi\right)} \\
\text{s.t.} & \sum\limits_{c\in\mathcal{C}} x_{jc} = 1 & \forall j\in\mathcal{J}, & \\
& x_{ic}+x_{jc} \leq 1 & \forall c\in\mathcal{C},\ \forall\{i,j\}\in E, & \\
& x_{ic}+x_{jc} \leq 1 + y_{ij} & \forall c\in\mathcal{C},\ \forall\{i,j\}\in \bar{E}, & \\
& x_{jc}\in\{0,1\} & \forall j\in\mathcal{J},\ \forall c\in\mathcal{C}, & (3.10) \\
& y_{ij}\in\{0,1\} & \forall \{i,j\}\in\bar{E}, & \\
& y_{ij} = 0 & \forall\{i,j\}\in E, & \\
& \sum\limits_{i\in\mathcal{J}}\sum\limits_{k=i+1}^{j_{i\pi}-1} y_{ik} - a \leq d_\pi & \forall\pi\in\mathcal{S}, & \\
& 0 \leq d_\pi & \forall \pi\in\mathcal{S} &
\end{array}$$

For various choices of parameter $\lambda > 0$ we will get efficient mean-risk schedules.

3.3 CVaR of number of unprocessed jobs

In this chapter we will apply the methodology from the last chapter to the problem of minimizing number of jobs that are not done which was presented in section 1.3. As it was mentioned before number of unprocessed job is better linked with possible penalty than number of overlaps. We will work again with the same set of scenarios as in the last chapter. For each pair of jobs and each scenario we will define a new binary parameter $g_{ij\pi}$ which indicates whether jobs i and j overlap under scenario π. Since this problem is based on problem (1.9) we will assume that the random delays of the jobs are independent as it was assumed before.

$$
\begin{array}{lll}
\min\limits_{x,y,z,a,d} & a + \dfrac{1}{1-\alpha}\sum\limits_{\pi\in\mathcal{S}} p_\pi d_\pi & \\
\text{s.t.} & \sum\limits_{c\in\mathcal{C}} x_{jc} = 1 & \forall j \in \mathcal{J}, \\
& x_{ic} + x_{jc} \le 1 & \forall c \in \mathcal{C}, \forall \{i,j\} \in E, \\
& x_{ic} + x_{jc} \le 1 + y_{ij} & \forall c \in \mathcal{C}, \forall \{i,j\} \in \bar{E}, \\
& y_{ij} + y_{ik} \le 1 + y_{jk} & \forall i,j,k \in \mathcal{J}, \\
& x_{jc} \in \{0,1\} & \forall j \in \mathcal{J}, \forall c \in \mathcal{C}, \\
& y_{ij} \in \{0,1\} & \forall \{i,j\} \in \bar{E}, \\
& y_{ij} = 0 & \forall \{i,j\} \in E, \\
& z_{j\pi} = \sum\limits_{i:\{i,j\}\in\bar{E}} z^*_{ij\pi} & \forall j \in \mathcal{J}, \forall \pi \in \mathcal{S}, \\
& z^*_{ij\pi} \ge 0 & \forall \{i,j\} \in \bar{E}, \forall \pi \in \mathcal{S}, \\
& 1 + z^*_{ij\pi} \ge (1 - z_{i\pi}) g_{ij\pi} + y_{ij} & \forall \{i,j\} \in \bar{E}, \forall \pi \in \mathcal{S}, \\
& z^*_{ij\pi} \le y_{ij} & \forall \{i,j\} \in \bar{E}, \forall \pi \in \mathcal{S}, \\
& z^*_{ij\pi} \le (1 - z_{i\pi}) g_{ij\pi} & \forall \{i,j\} \in \bar{E}, \forall \pi \in \mathcal{S}, \\
& \sum\limits_{i\in\mathcal{J}} z_{i\pi} - a \le d_\pi & \forall \pi \in \mathcal{S}, \\
& 0 \le d_\pi & \forall \pi \in \mathcal{S}
\end{array}
\tag{3.11}
$$

The notation used in this formulation is similar as for problem (1.9) with addition of scenario based decision variables d_π which have in this case meaning $max(0, \#unprocessed - a)$. Another difference is that in this formulation the decision variables z are scenario based and binary. (However for computational purposes they can be relaxed to real variables, they attain values only 0 or 1). In problem (1.9) they represented probability that job is not done or probability that job is not done because of delay of specific previous job. Here in problem (3.11) they are binary variables that indicate whether is that job unprocessed under scenario π and whether is that job unprocessed because of specific previous job.

To be more specific $z_{j\pi} = 1$ means that in scenario π job j is unprocessed and $z_{j\pi} = 0$ means that job j is processed. Also $z^*_{ij\pi} = 1$ means that under scenario π job i causes job j to be unprocessed. This can happen only if job i is processed ($z_{i\pi} = 0$), jobs i and j are overlapping in this scenario ($g_{ij\pi} = 1$) and both jobs are scheduled to the same machine ($y_{ij} = 1$).

At the end of section 1.3 we presented generalisation where we are looking for schedule with minimized expected penalty where we were considering for each job a penalty $r_j > 0$ which has to be paid if job j is not done. This generalization can be incorporated into problem (3.11) by simply changing second to last constraint into

$$\sum_{i\in\mathcal{J}} z_{i\pi} r_i - a \le d_\pi \qquad \forall \pi \in \mathcal{S}$$

Disadvantage of scenario based methods for minimizing $CVaR$ proposed in this thesis is that there is too many scenarios. Since each scenario is uniquely defined for each job by one of his consecutive jobs the number of scenarios is $J!$. That means the number of constraints and also the number of parameters and variables in this formulations is of order $J!$. For large number of jobs the problem is computationally not possible to solve. In the next chapter we will perform a short numerical study where we will find out what are the computational limitations for this problem.

3.3.1 Example problem

For illustration we will show a short example of the FIS problem. In this example we will work with $C = 3$ machines and $J = 5$ jobs. For probabilities of overlap between jobs i and j where $i < j$ we have:

$$\begin{array}{llll} p_{12} = 0.071 & p_{13} = 0.026 & p_{14} = 0.008 & p_{15} = 0.007 \\ & p_{23} = 0.210 & p_{24} = 0.058 & p_{25} = 0.057 \\ & & p_{34} = 0.082 & p_{35} = 0.081 \\ & & & p_{45} = 1 \end{array}$$

From the overlap probabilities we can see that only jobs 4 and 5 are always overlapping and therefore they can not be scheduled to the same machine. For all other jobs it means that if no delays occur then job 1 ends before job 2 begins, job 2 ends before job 3 begins and job 3 ends before job 4 begins. In terms of starting times s_j and prescribed finishing times f_j^0 we have:

$$s_1 < f_1^0 < s_2 < f_2^0 < s_3 < f_3^0 < s_4 < s_5 < f_4^0 < f_5^0.$$

The basic scheduling algorithm would be to schedule each job to the machine that was idle the longest time. In this example that would mean we will schedule job 1 to the machine 1, job 2 to the machine 2, job 3 to the machine 3, job 4 to the machine 1 and job 5 to the machine 2.

Job 4 will be unprocessed with probability $p_{14} = 0.008$ and job 5 will be unprocessed with probability $p_{25} = 0.057$.

The distribution of number of unprocessed jobs is:

- 0 jobs: $(1 - p_{14})(1 - p_{25}) = 0.935456$
- 1 job: $(1 - p_{14})p_{25} + p_{14}(1 - p_{25}) = 0.064088$
- 2 jobs: $p_{14}p_{25} = 0.000456$

Using formula (2.4) we will compute $CVaR_{0.95}$ as

$$\frac{(0.935456 + 0.064088 - 0.95) * 1 + 0.00456 * 2}{1 - 0.95} = 1.00912$$

In an alternative schedule where we switch machines for jobs 4 and 5, that means job 4 will be scheduled to machine 2 and job 5 to machine 1 we will have job 4 unprocessed with probability $p_{24} = 0.058$ and job 5 will be unprocessed with probability $p_{15} = 0.07$.

The distribution of number of unprocessed jobs for this schedule is:

- 0 jobs: $(1 - p_{15})(1 - p_{24}) = 0.935406$
- 1 job: $(1 - p_{15})p_{24} + p_{15}(1 - p_{24}) = 0.064188$
- 2 jobs: $p_{15}p_{24} = 0.000406$

and $CVaR_{0.95}$:

$$\frac{(0.935406 + 0.064188 - 0.95) * 1 + 0.000406 * 2}{1 - 0.95} = 1.00812$$

There we can see that the alternative schedule is slightly better with respect to $CVaR_{0.95}$ value. Computing problem (3.11) we will see that this schedule is the optimal schedule for this problem, that means it is the schedule with the lowest expected number of unprocessed jobs in the worst 5% of outcomes.

4. Numerical study

In this chapter we will perform a numerical study where we will try to minimize *CVaR* of number of unprocessed jobs for a few instances of scheduling problems. The formulation we will use is problem (3.11) from the last section of previous chapter. After that we will look at a few instances where we will perform the *mean-variance* optimization presented in problem (3.2). The simulation was conducted using *python 3.10.9* in *Jupyter Notebook*. The package used for optimization was *gurobipy* which worked with *Gurobi Optimizer version 11.0.1 build v11.0.1rc0*. An academic license was used for accessing full version of Gurobi Optimizer. Computer used to run the optimization uses *Windows 10.0* operating system. Processor in this computer is *AMD Ryzen 5 2500U* with *4 cores* and *2 GHz* and an *8 GB RAM*.

4.1 Data generation

Firstly we had to generate the problem instances. For that we started by defining parameters of the problem. Number of jobs - J, number of machines - C, number of problem instances we want to generate - N. For number of jobs we chose $J = 5$, $J = 6$, $J = 7$ and $J = 8$ jobs. For number of machines we chose $C = 3$ for all models. For all 4 of the types of problems we generated $N = 10$ instances.

The instances were simulated by generating starting time and duration of each of the J jobs. The starting times were generated by such way that differences between two consecutive starting times are independent identically distributed random variables with exponential distribution with scale parameter β_1. Also durations of the jobs are independent identically distributed with exponential distribution with scale parameter β_2. The random variables were generated using *numpy* library with seed specified before the first random variable was generated.

The parameters β_1 and β_2 we chose for our simulations are:

$$\beta_1 = 1 \qquad\qquad \beta_2 = 1$$

For each of the simulated instances we firstly checked if there exists feasible schedule using C machines. This check was performed by solving problem (1.1) where we checked for the chromatic number of the graph (smallest number of machines to perform the schedule) and compare the number with number of available machines C. We repeated generating new instances until we had N instances with feasible schedules.

Next thing that needs to be specified is distribution of random delays. As required by the assumptions of problem (3.11) the random delays are independent. We also assumed that the delays are identically distributed where the distribution is as follows. With probability p_{nd} no delay appears, with the complementary probability $(1 - p_{nd})$ the delay follows exponential distribution with scale parameter β_3.

The cumulative distributional function of the delay is therefore:

$$F_{D_j}(x) = \mathcal{P}(D_j(\xi) \leq x) = p_{nd} + (1 - p_{nd})(1 - e^{-x/\beta_3}),\ x \geq 0.$$

What interests us mostly is probability of two jobs overlapping and that can be calculated as

$$p_{ij} = \mathcal{P}(D_i(\xi) > s_j - f_i^0) = 1 - F_{D_i}(s_j - f_i^0) = (1 - p_{nd})e^{-(s_j - f_i^0)/\beta_3}$$

The parameters for the random delays p_{nd} and β_3 were chosen as:

$$p_{nd} = 0.75 \qquad\qquad \beta_3 = 1$$

4.2 Size of the optimization problem

Now let us look at the size of our problem, to be more specific number of variables and constraints that went into optimization solver. In the original formulation (3.11) we have 6 sets of variables.

- x_{jc} - $J*C$ binary variables indicating whether job j is scheduled to machine c.
- y_{ij} - $J*J$ binary variables indicating whether jobs i and j are scheduled to the same machine.
- $z_{j\pi}$ - $J*J!$ binary variables indicating whether job j is not processed under scenario π.
- $z^*_{ij\pi}$ - $J*J*J!$ binary variables indicating whether job j is not processed under scenario π because of delay of job i.
- d_π - $J!$ real-value variables that represent $max(0, \#unprocessed - a)$ under scenario π.
- a - single real-value variable that comes from the optimization formulation of $CVaR$.

Because of the constraint definition the $z_{j\pi}$ is just a sum of variables $z^*_{ij\pi}$ and therefore it can be relaxed to real-value variable. Also due to constraints the variable $z^*_{ij\pi}$ attains value $(1 - z_{i\pi})g_{ij\pi}y_{ij}$ which is either 0 or 1. Therefore also variables $z^*_{ij\pi}$ can be relaxed to real-value variables.

Altogether we have

- $J*(J+C)$ binary variables
- $J!*(J^2+J+1)+1$ real-value variables.

For the number of constraints we need to set $n_E = |E|$ and $\bar{n}_E = |\bar{E}|$. We will also not count the constraints which define x_{jc} and y_{ij} as binary variables because that is restriction on the variable type and not constraint. Also we added constraints $y_{ij} = y_{ji}$ and $y_{jj} = 1$ ($J^2 + J$ constraints) to ensure that the resulting matrix y is symmetric and clearly represents which pair of jobs is scheduled to the same machine.

Therefore the number of constraints is

$$J! * (2 + J + 4\bar{n}_E + n_E) + J^3 + J^2 + 2J + C * (n_E + \bar{n}_E).$$

We can clearly see that the biggest attribute to the size of the problem is $J!$ which represents number of scenarios we explore. Therefore in order to make the problem smaller we need to find a way to reduce number of scenarios. The way to do it is to throw away scenarios with probability $p_\pi = 0$. Some pairs of jobs overlap with probability 1 (pairs of jobs from set E) and therefore if $\{i, j\} \in E$ then j can not be the first job that job i does not overlap with. That means if $\{i, j\} \in \pi$ then $p_\pi = 0$ and we do not need to consider such scenario. Doing that we can lower the number of scenarios significantly.

Let us assume that the number of scenarios is $n_S \leq J!$ the number of real-value variables in our problem is then

$$n_S * (J^2 + J + 1) + 1$$

and number of constraints

$$n_S * (2 + J + 4\bar{n}_E + n_E) + J^3 + J^2 + 2J + C * (n_E + \bar{n}_E).$$

4.3 Optimization

To perform the optimization we needed to prepare some additional parameters. Firstly we needed to calculate scenario probabilities p_π. Those probabilities were calculated using equation (3.8). Next we prepared an array G of binary parameters $g_{ij\pi}$ which indicates whether job j starts before job i finishes under scenario π. Lastly we needed to specify lever α for witch we want to compute $CVaR_\alpha$. We decided for the common choice of $\alpha = 0.95$.

The optimization was performed by *Gurobi solver* with default setting. Time limit for the optimization was set to *3 hours*. Only 1 instance from the 8 jobs problem did not meet this time limit and therefore only best solution that was computed by this time is reported.

In the following tables (4.1), (4.2), (4.3) and (4.4) we can see results of the optimization for all 40 instances. The reported values are "*CVaR*" as optimal value of $CVaR_\alpha$ of number of unprocessed jobs, "*Exp*" as expected value of number of unprocessed jobs in the resulting schedule, "*gap*" represents relative gap between lower bound of objective function and value of objective function of optimal solution. If it is not equal to 0 that means the final schedule might not be the optimal one but that it lies in the gap. Default setting in *Gurobi* is that for gap smaller than 10^{-4} it stops the optimization process, that is why in the instance 2 of 7 job problem we have non zero gap. Next we have "*status*" which attains value 2 if the optimization algorithm finished and returned optimal solution and 9 if it exceeded time limit. Another reported values are "*runtime*" which is duration of optimization in seconds and "*scen*" which reports number of scenarios n_S for each instance. Last two reported values are "*vars*" which is number of variables and "*constr*" which means number of constraints.

	CVaR	Exp	gap	status	runtime	scen	vars	constr
0	1.214	0.207	0	2	0.040	16	537	613
1	1.265	0.244	0	2	0.050	32	1033	1155
2	0.352	0.018	0	2	0.157	120	3761	4511
3	1.086	0.160	0	2	0.063	20	661	794
4	1.061	0.143	0	2	0.031	24	785	868
5	1.011	0.062	0	2	0.094	40	1281	1473
6	0.615	0.031	0	2	0.078	90	2831	3252
7	1.008	0.065	0	2	0.078	60	1901	2232
8	1.021	0.140	0	2	0.091	96	3017	3457
9	0.378	0.019	0	2	0.156	120	3761	4511

Table 4.1: Results of 5 jobs problems.

	CVaR	Exp	gap	status	runtime	scen	vars	constr
0	2.054	0.380	0	2	0.062	72	3151	3054
1	1.453	0.269	0	2	0.047	64	2807	2877
2	1.691	0.334	0	2	0.547	288	12439	13561
3	1.015	0.118	0	2	0.157	90	3925	4274
4	1.022	0.058	0	2	0.227	192	8311	9145
5	1.021	0.063	0	2	0.630	300	12955	14713
6	1.278	0.221	0	2	0.116	72	3151	3339
7	1.233	0.193	0	2	0.638	240	10375	11832
8	1.729	0.361	0	2	0.485	192	8311	8762
9	1.037	0.111	0	2	0.537	360	15535	16874

Table 4.2: Results of 6 jobs problems.

	CVaR	Exp	gap	status	runtime	scen	vars	constr
0	1.319	0.252	0	2	3.625	2000	114071	126475
1	1.021	0.057	0	2	7.911	1944	110879	122947
2	1.095	0.119	0	2	59.594	3780	215531	261291
3	1.227	0.189	0	2	31.322	2520	143711	169312
4	2.518	0.726	0	2	1.592	480	27431	27838
5	2.592	0.705	8e-5	2	0.118	540	30851	30179
6	2.110	0.410	0	2	8.585	960	54791	57117
7	1.055	0.091	0	2	9.044	2016	114983	131513
8	1.066	0.095	0	2	10.210	1200	68471	76075
9	0.888	0.044	0	2	50.708	4032	229895	278679

Table 4.3: Results of 7 jobs problems.

In table (4.5) we can see average of number of scenarios and average of runtime for each number of jobs. These data can be seen plotted on logarithmic scales in Figure 4.1. The left axis with the red line represent dependence of average number of scenarios on the number of jobs. The right axis with the blue line represent dependence of average runtime on number of jobs. We can see that both of them are growing faster than exponentially. (They are growing with $J!$)

	CVaR	Exp	gap	status	runtime	scen	vars	constr
0	1.199	0.191	0	2	126.736	15552	1135385	1291499
1	1.184	0.163	0	2	8329.176	30240	2207609	2692040
2	1.253	0.210	0	2	10329.384	29400	2146289	2617280
3	1.925	0.358	0	2	96.163	7200	525689	583883
4	2.251	0.554	0.137	9	10800.728	18816	1373657	1637673
5	1.763	0.300	0	2	487.300	10080	735929	837323
6	1.073	0.110	0	2	216.742	29400	2146289	2558481
7	1.103	0.111	0	2	1043.308	11200	817689	975080
8	1.537	0.283	0	2	1460.843	11520	841049	956842
9	2.497	0.676	0	2	34.953	4200	306689	332485

Table 4.4: Results of 8 jobs problems.

J	scenarios	runtime
5	61.8	0.0838
6	187	0.3446
7	1947.2	18.2709
8	16760.8	3292.5333

Table 4.5: Average number of scenarios and runtime for different number of jobs.

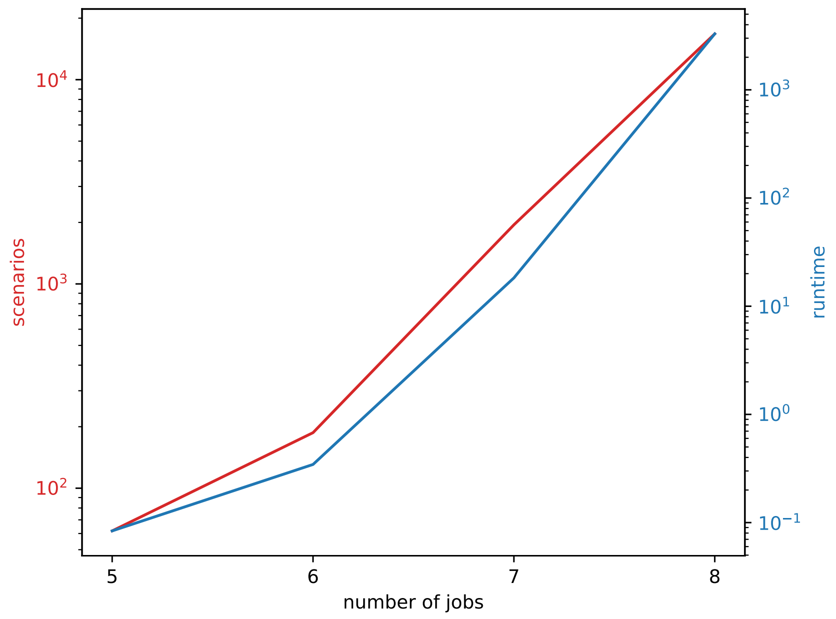

Figure 4.1: Average number of scenarios and runtime for different number of jobs.

For higher number of jobs the optimization problem was not possible to solve. For $J = 9$ the computer raised out of memory error at the beginning of the optimization. Using a different computer with better computing power and bigger RAM the problem might be solvable.

4.4 Mean-variance model

Second model we decided to implement is mean-variance optimization of number of overlaps. We considered the delays of the jobs to be independent, therefore we used linear model (3.2) for optimization. We managed to increase number of jobs in the schedule to $J = 25$ and number of machines to $C = 5$. Other parameter were chosen the same as in previous simulation.

For data generation we used the same process as described in section 4.1. The size of the problem was also significantly different. In this problem we used only 3 sets of variables:

- x_{jc} - $J*C$ binary variables indicating whether job j is scheduled to machine c.
- y_{ij} - $\bar{n}_E$ binary variables indicating whether jobs i and j are scheduled to the same machine. Different than in previous problem is that we need this variable only for the pairs of jobs that can be scheduled to the same machine.
- z_{ijk} - $\bar{n}_E * \bar{n}_E$ binary variables indicating whether triplet of jobs i, j and k is scheduled to the same machine. Since $\bar{n}_E$ is of order J^2 we have of order J^4 variables z. But since in optimization we will actively use only those that share the same job on the second place of first pair $\{i, \mathbf{j}\} \in \bar{E}$ and first place of second pair $\{\mathbf{j}, k\} \in \bar{E}$ the number of active variables is of order J^3.

For number of constraints it is not easy to compute exactly because it depends on the number of actively used variables z_{ijk}, but it is possible to compute upper bound. By actively used variable we mean variable that is used in at least one constraint. For actively used variable z_{ijk} it holds that $1 \leq i < j < k \leq J$. That gives us upper bound of $J * (J-1) * (J-2)/6$ actively used variables. Therefore the number of constraints is bounded by:

$$J + C * (n_E + \bar{n}_E) + \frac{1}{6} * J * (J-1) * (J-2)$$

It also holds that $n_E + \bar{n}_E = J * (J-1)/2$ since it is the number of all pairs of jobs, therefore the upper bound of number of constraint is:

$$\frac{C * J^2}{2} - \frac{C * J}{2} + \frac{J^3}{6} - \frac{J^2}{2} + \frac{4J}{3}$$

We can see that the number is growing with J^3 which is significantly slower than $J!$ in previous problem. For our choice of parameters ($J = 25$, $C = 5$) the upper bound is 3825 constraints.

For the optimization we did not need to compute more input parameters, we needed only the probabilities of pair of job overlapping. Parameters v_{ij} and v_{ijk} in the objective function are only direct transformation of those probabilities therefore they did not need to be precomputed. Last parameter used in the objective function λ had to be chosen. For our simulation we decided for $\lambda = 0.5$.

	var	Exp	gap	status	runtime	vars	constr
0	5.537	0.439	0	2	28.453	79931	3437
1	4.925	0.212	0	2	131.974	79931	3439
2	3.826	0.305	0	2	48.478	73837	3221
3	4.121	0.099	0	2	12.103	84515	3603
4	3.896	0.240	0	2	11.503	78805	3398
5	5.100	0.319	0	2	77.216	77131	3325
6	5.033	0.317	0	2	224.559	79367	3407
7	3.604	0.040	0	2	29.437	82781	3535
8	5.918	0.611	0	2	29.014	74927	3260
9	3.719	0.208	0	2	8.864	78245	3381

Table 4.6: Results of mean-variance problem.

In table (4.6) we can see results of the simulation. The reported value of "*var*" is variance of number of overlaps in the final schedule, also "*Exp*" represents expected number of overlaps in the final schedule. All other columns are the same as in previous tables. We can see that in all instances the algorithm successfully computed optimal schedule.

For larger schedules with more than 25 jobs the computing time started to grow rapidly. With time limit set to *1 hour*, the highest instance we were able to solve was with $J = 30$ jobs.

Conclusion

In this thesis we discussed fixed interval scheduling problem with random delays. In the first chapter we presented what the fixed interval scheduling means and what are some common approaches to the problem. Specifically maximizing reliability of the schedule and minimizing number of overlaps between the jobs. At the end of the first chapter we introduced a new approach to the problem which was minimizing expected number of unprocessed jobs.

The second chapter contains a short introduction into risk measures. We discuss some properties and usage of risk measures and coherent risk measures. Afterwards we present common risk measures which are used in the third chapter. Specifically we focused on variance and conditional value at risk.

In the third chapter we combine fixed interval scheduling problem with risk measures. The aim was to create new formulations of FIS problem where risk is taken into account. At first we created a new mean-variance optimization problem where the resulting optimal schedule is efficient considering bi-criteria problem where we want to minimize both expected value and variance of number of overlaps between jobs. The second approach was to minimize conditional value at risk of number of overlaps and lastly we presented a new formulation where we minimized conditional value at risk of number of unprocessed jobs.

In the last chapter we performed a numerical study where we tried computational limits for our newly introduced formulations from third chapter. For the minimization of $CVaR$ of number of unprocessed we found out that it is possible to solve only for small instances because the the complexity of the problem rises very fast with number of jobs. Therefore for bigger instances we decided to implement also mean-variance optimization which was able to compute optimal schedule for much bigger instances.

Bibliography

M. Branda. Distributionally robust fixed interval scheduling on parallel identical machines under uncertain finishing times. *Computers & Operations Research*, 98:231–239, 2018. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2018.05.025.

M. Branda and Š. Hájek. Flow-based formulations for operational fixed interval scheduling problems with random delays. *Computational Management Science*, 14:161–177, 2017.

M. Branda, J. Novotný, and A. Olstad. Fixed interval scheduling under uncertainty – a tabu search algorithm for an extended robust coloring formulation. *Computers & Industrial Engineering*, 93:45–54, 2016. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2015.12.021. URL `https://www.sciencedirect.com/science/article/pii/S036083521500501X`.

H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 00221082, 15406261. URL `http://www.jstor.org/stable/2975974`.

A. J. McNeil and J. Nešlehová. Multivariate Archimedean copulas, d-monotone functions and l1-norm symmetric distributions. *The Annals of Statistics*, 37 (5B):3059 – 3097, 2009. doi: 10.1214/07-AOS556. URL `https://doi.org/10.1214/07-AOS556`.

R. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26:1443–1471, 07 2002. doi: 10.1016/S0378-4266(02)00271-6.

R. Rockafellar, S. Uryasev, and M. Zabarankin. Master funds in portfolio analysis with general deviation measures. *Journal of Banking & Finance*, 30(2):743–778, 2006. ISSN 0378-4266. doi: https://doi.org/10.1016/j.jbankfin.2005.04.004. URL `https://www.sciencedirect.com/science/article/pii/S0378426605000920`. Risk Management and Optimization in Finance.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming. Modeling and theory.* 01 2009. doi: 10.1137/1.9780898718751.