Quality recovering of university timetables

Lindahl, Michael; Stidsen, Thomas Jacob Riis; Sørensen, Matias

Published in:
European Journal of Operational Research

Link to article, DOI:
10.1016/j.ejor.2019.01.026

Publication date:
2019

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Lindahl, M., Stidsen, T. J. R., & Sørensen, M. (2019). Quality recovering of university timetables. *European Journal of Operational Research*, *276*(2), 422-435. https://doi.org/10.1016/j.ejor.2019.01.026

Quality Recovering of University Timetables

Michael Lindahl[a,b,*], Thomas Stidsen[a], Matias Sørensen[b]

[a] *Department of Management Engineering*
Technical University of Denmark
[b] *MaCom A/S*
Copenhagen

Abstract

At universities, the timetable plays a large role in the daily life of students and staff, showing when and where lectures are given. But whenever a schedule is executed in a dynamic environment, disruptions will occur. It is then desirable to find a new timetable similar to the old one, so only a few people will be affected. This leads to a minimum perturbation problem, where the goal is to find a feasible timetable by changing as few assignments as possible.

In this paper we show that minimum perturbation solutions often have low quality and how using additional perturbations results in timetables with significantly higher quality while still keeping the number of perturbations low.

We formulate a bi-objective model and propose a method to solve it by using mixed integer programming. We test the method on standard instances of the Curriculum-based Course Timetabling Problem with four different types of disruptions. This allows the decision makers to determine the best trade-off between the number of perturbations and the quality, ultimately leading to better timetables for students and staff when disruptions occur.

Keywords: Timetabling, Disruptions, Multiple objective programming, Minimum Perturbation, Integer programming

1. Introduction

Disruptions are unavoidable to all schedules executed in a dynamic environment. This is also the case at universities where the timetable determines when and where lectures are taught. After a timetable is finalized and published to lecturers and students, changes will inevitably occur. For example, a lecturer might become unable to teach at a certain time during the week, or a room might get reserved for a conference for an entire day.

When disruptions occur to a published timetable, changing the timetable will cause inconvenience for the effected parties. Therefore, it is desirable that the new timetable is as similar to the old one as possible, so the changes do not affect too many people. This leads to the

[*] corresponding author

Email addresses: `miclindahl@gmail.com` (Michael Lindahl), `thst@dtu.dk` (Thomas Stidsen), `sorensen.matias@gmail.com` (Matias Sørensen) minimum perturbation problem where the goal is to reconstruct feasibility for an infeasible schedule by making as few perturbations (changes) as possible.

However, creating a feasible timetable is only a part of the goal of the minimum perturbation problem. The quality of the timetable is of great importance to universities. The quality is defined by soft constraints, which are undesired features of the timetable that should be avoided. Variations of these features that occur in university timetabling are described in Bonutti et al. (2012). Using the solution with the minimum number of perturbations lead to a big increase in the violation of the soft constraints given. This is undesirable, and it is likely that the planner wants to make additional perturbations to obtain a timetable of higher quality.

Example. In table 1, a small example of how a minimum perturbation solution can be improved with an additional perturbation. The disrupted timetable shows the timetable for a day where Room B becomes unavailable at 15:00 when there is supposed to be a Chem lecture. A feasible solution can be obtained by moving the lecture to the empty Room A in the previous timeslot. This room is, despite being feasible, undesired by the Chem teacher as she already has another chem lecture in Room B on that day and has some equipment she needs to use for both lectures. But by swapping rooms with the Algo class, which does not care much about which room to use, the Chem teacher becomes happy with very little sacrifice from the Algo teacher.

Disrupted timetable

	Room A	Room B
09:00	Mech	Chem
11:00	Math	Engl
13:00	-	Algo
15:00	Geo	⚠Chem

Minimum perturbation

	Room A	Room B
09:00	Mech	Chem
11:00	Math	Engl
13:00	☹ *Chem*	Algo
15:00	Geo	⚠

One extra perturbation

	Room A	Room B
09:00	Mech	Chem
11:00	Math	Engl
13:00	☺ *Algo*	☺ *Chem*
15:00	Geo	⚠

Table 1: An example of a disrupted timetable, where Room B becomes unavailable at 15:00. The minimum perturbation solution is moving a single lecture, but will require the lecture to use a room that is usable but is not desirable for that specific lecture. By swapping rooms with another lecture, it leads to a better timetable, with little extra inconvenience for the invovled parties.

The purpose of this paper is to analyze the trade-off between perturbations and quality. , and how the addition of more perturbations will improve the quality. Providing planners with these trade-offs can assist them in recovering an infeasible timetable and still obtain high-quality timetables.

The paper is organized as follows: In Section 2 we describe the perturbation problem and how we model it. In Section 3 we show an algorithm to solve the problem. In Section 4 we show the computational results, and finally, in Section 5 we give our conclusions.

1.1. *Previous work*

Minimum perturbation problems in university timetabling have received little attention in the literature compared to the full static problem of creating the entire timetable. The first work on minimum perturbations in scheduling was in Sakkout and Wallace (2000) who presented an algorithm based on constraint programming to minimally reconfigure a timetable that has become invalid. Later, Elkhyari et al. (2003) also use constraint programming to solve timetabling problems modeled as a minimum perturbation problem on Resource-Constrained Project Scheduling Problems. Zivan et al. (2011) propose a hybrid search, also using constraint programming to solve minimum perturbation problems for scheduling. An improved algorithm, utilizing lower bounds, was proposed in Fukunaga (2013).

The first use of the minimum perturbation approach on real university timetabling instances was by Barták et al. (2003), where they propose a branch-and-bound-like algorithm to find approximate solutions. This work is extended by Müller et al. (2005), where they modify the iterative forward search for the static problem to solve minimum perturbation problems. They preserve quality as a weight in the objective when searching and show the correlation between the size of the disruption and the impact on quality. Finally, Rudová et al. (2011) describe their whole system for timetabling at a large university that both incorporates minimal perturbation problems and also problems they call *interactive problems*, where the software suggests small changes to existing timetables.

More recently, Phillips et al. (2016) use mixed integer programming to solve the minimum perturbation problem on real life instances from the University of Auckland. To avoid solving large models they create a smaller problem with only a part of the timetable, and if no feasible solution is found, they gradually expand it until a valid perturbation is found. Mixed-integer programming has however been used on the static university timetabling, see for example Ferland and Roy (1985); Burke et al. (2008); Lach and Lübbecke (2008); Phillips et al. (2015). Using a MIP solver to explore a neighborhood around a solution has also been used to create heuristics, such as Local Branching by Fischetti and Lodi (2003) where a constraint on the hamming distance is added to the problem to create smaller problems that are easier to solve.

For a more broad perspective on dynamic problems we refer to Kocjan (2002) and Verfaillie and Jussien (2005).

Overall, there is a lack of research that examines the loss of quality when choosing the minimal perturbation solution, and what the potential benefit is when using additional perturbations. Mixed Integer Programming (MIP) has only been used in one previous paper to solve this problem, but we believe that MIP models and their solvers are well suited.

2. The quality recovering problem

A perturbation problem consists of three components which are described in this section. In loose terms, the perturbation problem can be defined as follows,

$$\textit{Pertubation problem} = \textit{static problem} + \textit{solution} + \textit{disruption} + \Delta$$

The first part is the static problem, which is the underlying problem formulation needed to create the timetable in the first place (containing all the hard and soft constraints). The second part is the solution i.e., a feasible assignment of the courses with respect to the static formulation. The third part is a disruption that changes the formulation and makes the solution infeasible. Finally, to calculate the similarity between two solutions a perturbation function is used, denoted by Δ. The perturbation function calculates how many changes are made to the initial solution to reach the new solution.

In the quality recovering problem we will distinguish between four solutions defined as follows:

- S_{Init}: The initial solution that after the disruption becomes infeasible.
- S_{QR}^{Δ}: The best feasible solution with the distance Δ from the initial solution. Two special cases of this solution are:
  - $S_{\text{QR}}^{\min}$: The minimum perturbation solution i.e. the closest feasible solution, where min is the number of perturbations from the initial solution.
  - $S_{\text{QR}}^{\max}$: The solution with the best quality, where max is the number of perturbations from the initial solution.

Figure 1 illustrates the relation between these solutions, the solution space and the disruption. In the following we describe the specific parts of the perturbation problem for the university course timetabling problem.

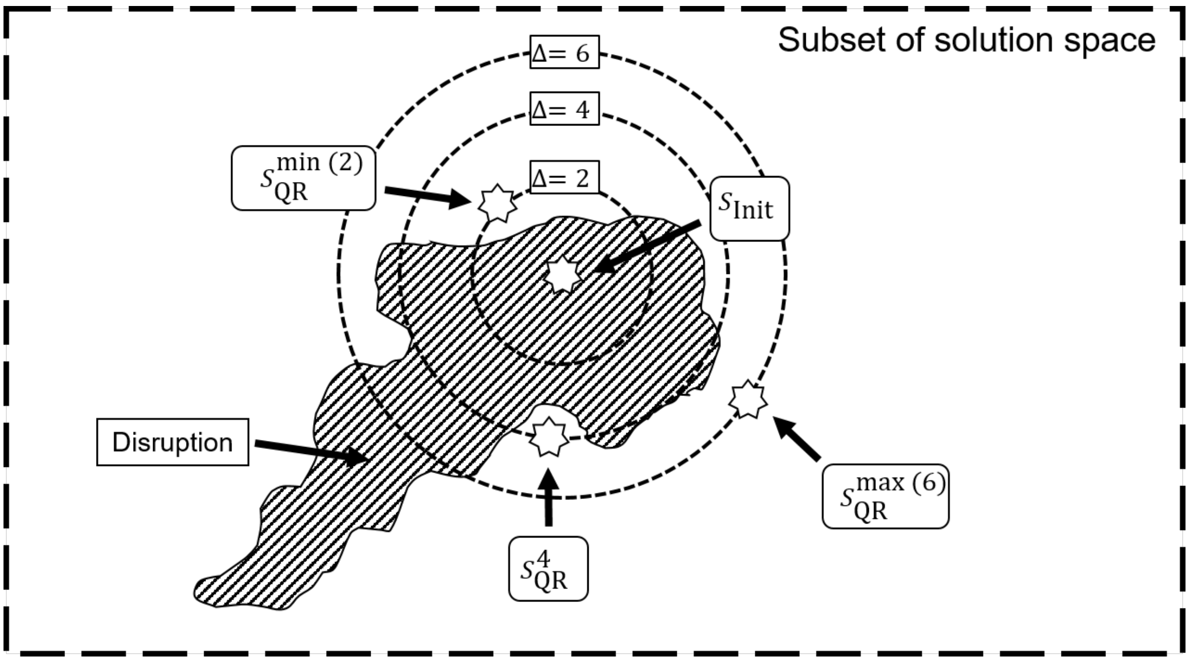

Figure 1: An example of the perturbation problem. The figure shows a subset of the solution space. The disruption makes a subspace of the solution space infeasible (grey area), including the initial solution. The circles show three distances from the initial solution. The minimum perturbation solution is the nearest feasible solution to the initial solution. The quality recovering solution is the one that is further away but has higher quality.

2.1. *Static problem*

The static problem comes from the Second International Timetabling Competition ITC-2007 stated in Di Gaspero et al. (2007), namely Curriculum-based Course Timetabling. The problem was created to allow researchers to compare algorithms and results on the same instances. The problem consists of assigning lectures to timeslots and rooms while taking both hard and soft constraints into account.

All instances are real-world data sets from the University of Udine. The hard and soft constraints originate from this university, but were generalized with the goal of not making the model too complex while still having different types of constraints to capture the essence of real world timetabling. This problem has received much attention in the literature as the de-facto benchmarking data set for university timetabling. For an overview of the literature on this, we refer to the survey by Bettinelli et al. (2015).

First, we will describe the problem and explain the model by using mixed-integer programming. Burke et al. (2008) proposed the first model known as the monolithic model or the three-index model, seen in Model 1. First, the hard constraints given in (1e)-(1h) are described and then the soft constraints given in (1i)-(1m).

2.1.1. *Hard constraints*

In curriculum-based course timetabling there is given a list of courses, $\mathscr{C}$, a list of timeslots, $\mathscr{P}$, and a list of rooms, $\mathscr{R}$. A course $c \in \mathscr{C}$ consists of a number of lectures, $L(c)$. The goal is then to schedule all lectures by assigning them to a timeslot and a room. For this we use the following binary decision variable that determines which assignments are used.

$$x_{c,p,r} = \begin{cases} 1 & \text{if course } c \in \mathscr{C} \text{ is planned at period } p \in \mathscr{P} \text{ and in room } r \in \mathscr{R} \\ 0 & \text{otherwise} \end{cases}$$

First of all, every lecture needs to be scheduled, which is handled by constraint (1e). A room can only fit one course in it, which is ensured by constraint (1f). We also have a set of teachers $\mathscr{T}$, where $\mathscr{C}(t)$ is the set of courses taught by teacher $t \in \mathscr{T}$. A teacher can only teach one course at the time ensured by constraint (1g). Furthermore, students can also only be at one place at a time. Therefore, a set of curricula $\mathscr{CU}$ is given and courses that are a part of the same curriculum, $\mathscr{C}(cu)$, cannot be placed in the same timeslot, ensured by constraint (1h). Last, there are unavailabilities, which are timeslots to which specific courses cannot be assigned. We will set $x_{c,p,r} = 0$ if course $c \in \mathscr{C}$ cannot be taught in timeslot $p \in \mathscr{P}$.

2.1.2. *Soft constraints*

The quality of a timetable is measured by how many soft constraints are being violated. In total of soft constraints are defined, and each one is associated with a number of penalty points. The objective function (1a)-(1d) is then equal to the sum of these penalty points. The soft constraints are the following:

RoomCapacity. Each room is associated with a capacity, $cap(r)$, and each course is associated with a number of students attending the course, $dem(c)$. A penalty of *1* is given for each extra student assigned to a room. This is calculated in the objective (1a), where we set the cost equal

$$
\begin{aligned}
\min \quad & f_{qual} = \sum_{c\in\mathscr{C},p\in\mathscr{P},r\in\mathscr{R}} obj(c,r)\cdot x_{c,p,r} && && \text{(1a)}\\
& + \sum_{c\in\mathscr{C},r\in\mathscr{R}} 1\cdot y_{c,r} - |\mathscr{C}| && && \text{(1b)}\\
& + \sum_{c\in\mathscr{C}} 5\cdot w_c && && \text{(1c)}\\
& + \sum_{cu\in\mathscr{CU},p\in\mathscr{P}} 2\cdot v_{cu,p} && && \text{(1d)}\\
\text{s.t.} \quad & \sum_{p\in\mathscr{P},r\in\mathscr{R}} x_{c,p,r} = L(c) && \forall c\in\mathscr{C} && \text{(1e)}\\
& \sum_{c\in\mathscr{C}} x_{c,p,r} \leq 1 && \forall p\in\mathscr{P}, r\in\mathscr{R} && \text{(1f)}\\
& \sum_{c\in\mathscr{C}(t),r\in\mathscr{R}} x_{c,p,r} \leq 1 && \forall t\in\mathscr{T}, p\in\mathscr{P} && \text{(1g)}\\
& \sum_{c\in\mathscr{C}(cu),r\in\mathscr{R}} x_{c,p,r} \leq 1 && \forall cu\in\mathscr{CU}, p\in\mathscr{P} && \text{(1h)}\\
& \sum_{p\in\mathscr{P}} x_{c,p,r} - |\mathscr{P}|\cdot y_{c,r} \leq 0 && \forall c\in\mathscr{C}, r\in\mathscr{R} && \text{(1i)}\\
& \sum_{p\in d,r\in\mathscr{R}} x_{c,p,r} - z_{c,d} \geq 0 && \forall c\in\mathscr{C}, d\in\mathscr{D} && \text{(1j)}\\
& \sum_{d\in\mathscr{D}} z_{c,d} + w_c \geq mnd(c) && \forall c\in\mathscr{C} && \text{(1k)}\\
& \sum_{c\in\mathscr{C}(cu),r\in\mathscr{R}} x_{c,p,r} - r_{cu,p} = 0 && \forall cu\in\mathscr{CU}, p\in\mathscr{P} && \text{(1l)}\\
& - r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} \leq 0 && \forall cu\in\mathscr{CU}, p\in\mathscr{P} && \text{(1m)}\\
& x_{c,p,r} \in \mathbb{B} && \forall c\in\mathscr{C}, p\in\mathscr{P}, r\in\mathscr{R} && \text{(1n)}\\
& y_{c,r} \in \mathbb{B} && \forall c\in\mathscr{C}, r\in\mathscr{R} && \text{(1o)}\\
& w_c \in \mathbb{Z}_+ && \forall c\in\mathscr{C} && \text{(1p)}\\
& z_{c,d} \in \mathbb{B} && \forall c\in\mathscr{C}, d\in\mathscr{D} && \text{(1q)}\\
& v_{cu,p} \in \mathbb{B} && \forall cu\in\mathscr{CU}, p\in\mathscr{P} && \text{(1r)}\\
& r_{cu,p} \in \mathbb{B} && \forall cu\in\mathscr{CU}, p\in\mathscr{P} && \text{(1s)}
\end{aligned}
$$

Model 1: The MIP model for the static university timetabling problem.

to the undercapacity of room r if course c is assigned to it i.e. $obj(c,r) = \max(0, dem(c) - cap(r))$.

RoomStability. All lectures from the same course are assigned to the same room. A penalty of *1* is given for each extra room used. To calculate this we introduce the binary variable $y_{c,r}$ to indicate if course c is scheduled in room r, as ensured by constraint (1i). The violation is then calculated in objective (1b) as the sum of rooms used minus the total number of courses.

MinimumWorkingDays. The workload of a course should be distributed throughout the week. Each course, c, therefore, has a number of minimum working days, $mnd(c)$, on which it at least should be planned. Let $\mathscr{D}$ be the set of days, and let $p \in d$ be all the timeslots p on day $d \in \mathscr{D}$. The binary variable $z_{c,d}$ indicates if course c is scheduled on day d, ensured by constraint (1j). The number of violation, w_c, for course c is then calculated by constraint (1k) and summed up in objective (1c) with a penalty of *5*.

CurriculumCompactness. Students should not have idle timeslots, and therefore it is desirable that a lecture from a given curriculum is next to a lecture from the same curriculum. *Two* penalty points are given for each violation. The binary variable $r_{cu,p}$ indicates if a lecture from curriculum cu is planned in timeslot p, ensured by constraint (1l). The variable $v_{cu,p}$ then indicates if curriculum cu has no neighbors in timeslot p, ensured by constraint (1m) and added in the objective in (1d).

2.2. Disruption

In this section we introduce the concept of disruption in terms of the model described in Section 2.1. Disruptions take different forms depending on the type of environment in which they occur, but are always related to a resource, e.g. a room, teacher, student or a timeslot.

In this paper we will consider two types of disruptions: 1) A resource is removed i.e. unavailable for assignment, and 2) a new shared resource is added, i.e. some courses share a resource and cannot be assigned to the same timeslot. These two very generic disruption types show that MIP can represent the different kinds of disruptions that occur at a university. A removed resource is for example when a room is unavailable or a course cannot be taught in a specific timeslot. This is modeled in the following way by altering the MIP model. Let J be the set of tuples with all combinations of courses, timeslots and rooms.

$$J = \{(c, p, r) : c \in \mathscr{C},\ p \in \mathscr{P},\ r \in \mathscr{R}\} \tag{2}$$

Then let $\hat{J} \subset J$ be the set of resource combinations that are removed and fix those variables to zero in the model, thereby making those solutions infeasible.

$$x_{c,p,r} = 0 \quad \forall (c, p, r) \in \hat{J} \tag{3}$$

Similar to removing a resource, a new shared one could also be added. This is for example the case if a new curriculum is added because a group of students needs to be able to take a specific set of courses. This will result in a conflict if these courses are taught at the same time. Let $\hat{\mathscr{C}} \subset \mathscr{C}$ be the courses that share the new resource. The following constraint is then added to the model,

$$\sum_{c \in \hat{\mathscr{C}}, r \in \mathscr{R}} x_{c,p,r} \leq 1 \quad \forall p \in \mathscr{P} \tag{4}$$

Both of the two disruption types restrict the static model. This means that a new solution, S^{Δ}_{QR}, can never be better than the initial solution, S_{Init}, as a feasible solution to the dynamic problem also is a feasible solution to the static problem, and the objective is the same.

$$
\begin{array}{lll}
\min & f_{qual} \quad \text{(as defined by (1a)-(1d))} & \text{(6a)} \\
 & f_\Delta = \Delta(x,\bar{x}) \quad \text{(as defined by (5))} & \text{(6b)} \\
\text{s.t.} & (1e) - (1s) & \text{(6c)} \\
 & \text{disruption} \quad \text{(either (3) or (4))} & \text{(6d)}
\end{array}
$$

Model 6: The bi-objective MIP model for the quality recovering problem, consisting of the static model with an additional objective and a disruption.

2.3. *Perturbation function*

To measure the amount of changes between the initial solution, S_{Init}, and a new solution, S_{QR}^{Δ}, we need a perturbation function that will be our second objective. For this we use the hamming distance to calculate the number of decision variables of which the value have changed. Because the sum of all variables is always the same, due to constraint (1e), we only calculate the values that change from one to zero. Let $\bar{J} = \{(c,p,r) \in J : \bar{x}_{c,p,r} = 1\}$. The hamming distance between two solutions are defined as,

$$\Delta(x,\bar{x}) = \sum_{(c,p,r)\in\bar{J}} (1 - x_{c,p,r}) \tag{5}$$

More advanced perturbation functions can be modelled by including the variables that switches from zero to one with a weight to describe how expensive they are. For example, this could model that it is more desirable to move a course to a different room instead of moving it to a different day.

2.4. *Quality recovering model*

Altering the original static model by adding a disruption and a new minimum perturbation objective results in the bi-objective model shown in Model 6. $\bar{x}$ is the solution from S_{Init}. All constraints and variables from the static problem are added in constraint (6c) and in the objective (6a). The perturbation measure is added as a second objective (6b). Finally, we add the disruption in constraint (6d). As mentioned, this disruption can take the different forms described in Section 2.2. The two solutions $S_{\text{QR}}^{\min}$, $S_{\text{QR}}^{\max}$ are equal to the two lexicographic solutions, i.e. minimizing the objectives in prioritized order.

3. Quality recovering algorithm

To solve the quality recovering problem in Model 6 we need to take both objectives into account; the quality objective f_{qual} and the perturbation objective f_Δ. These two objectives are conflicting, as keeping the number of perturbations down limits the potentially improving solutions that can be found. This results in a bi-objective optimization problem. To solve this we search for pareto-optimal solutions, which are defined as solutions where one objective cannot be improved without worsening the other.

To generate this solution frontier, we base our algorithm on the ϵ-constraint method from Haimes et al. (1971). The approach is to put a constraint on one of the objectives while minimizing the other and repeat this with a new constraint. The pseudo code is seen in Algorithm 3.1. The method starts by solving the minimum perturbation problem and then puts a constraint on the perturbation objective. It then iteratively increases the allowed number of perturbations while minimizing the quality objective.

The algorithm is simple and fully sufficient for our problem. Because it uses mixed integer programming underneath, it is very generic and can be used with most of static problems as well as different perturbation functions depending on the problem at hand.

Algorithm 3.1 Quality Recovering

```
1: $\tilde{\Delta} \leftarrow Minimize(f_\Delta)$                      ▷ Minimum Pertubations
2: $\tilde{f}_{qual} \leftarrow \infty$
3: repeat
4:     Update $\epsilon$-constraint: $\tilde{f}_\Delta = \tilde{\Delta}$
5:     $S_{\text{QR}}^{\tilde{\Delta}} \leftarrow Minimize(f_{qual})$                 ▷ Find Pareto-solution
6:     $\tilde{\Delta} \leftarrow \tilde{\Delta} + 1$
7: until Stopping criteria met
```

4. Computational results

To show the applicability of our method we use the data sets from the Second International Timetabling Competition, described in Di Gaspero et al. (2007). The 21 data sets can be seen in Table 2, and it is seen that there is a variance between , including number of curricula. We will use the currently best-known solution for each instance for the initial solution, S_{Init}, and then disrupt it so it becomes infeasible. All data sets and best-known solutions can be found at `http://tabu.diegm.uniud.it/ctt/`. The complete source-code to make the computations are available on `http://github.com/miclindahl/UniTimetabling`.

We will analyze the impact of four different disruptions that are very different, both in terms of the way they impact the timetable and in terms of the number of lectures affected. These disruptions cover scenarios that usually happen at a university. The four disruptions are the following:

One Assignment Invalid One assignment, $(\hat{c}, \hat{p}, \hat{r})$, is invalid and needs to be assigned to a different room or to a different timeslot. The following constraint is added,

$$x_{\hat{c},\hat{p},\hat{r}} = 0$$

Insert Curriculum Takes four courses $\hat{\mathscr{C}}$ and put them into a curriculum meaning that they cannot be taught at the same time. To be able to compare with S_{Init} the objective function is not changed. Only the following constraint is added,

$$\sum_{c \in \hat{\mathscr{C}}, r \in \mathscr{R}} x_{c,p,r} \leq 1 \quad \forall p \in \mathscr{P}$$

Instance	Courses	Rooms	Timeslots	Curricula
comp01	30	6	30	14
comp02	82	16	25	70
comp03	72	16	25	68
comp04	79	18	25	57
comp05	54	9	36	139
comp06	108	18	25	70
comp07	131	20	25	77
comp08	86	18	25	61
comp09	76	18	25	75
comp10	115	18	25	67
comp11	30	5	45	13
comp12	88	11	36	150
comp13	82	19	25	66
comp14	85	17	25	60
comp15	72	16	25	68
comp16	108	20	25	71
comp17	99	17	25	70
comp18	47	9	36	52
comp19	74	16	25	66
comp20	121	19	25	78
comp21	94	18	25	78

Table 2: The 21 instances used from the second international timetabling competition defined in Di Gaspero et al. (2007).

Remove Room Whole Day Removes a room $\hat{r}$ for an entire day $\hat{d}$, i.e. adding constraint:

$$x_{c,p,\hat{r}} = 0 \quad \forall c \in \mathscr{C}, p \in \hat{d}$$

One Timeslot Unavailable Makes one timeslot $\hat{p}$ unavailable for all courses. The following constraint is added,

$$x_{c,\hat{p},r} = 0 \quad \forall c \in \mathscr{C}, r \in \mathscr{R}$$

. All computations are made on a 64 bit Windows machine with a 4 GHz Intel Core i7 CPU and 32 GB of memory. To solve the integer programs we use Gurobi with standard settings.

$$\text{max perturbations} = \max(f_\Delta(S_{\text{QR}}^{\min}) + 5, 15) \tag{7}$$

We also add a timelimit on 10,800 seconds (3 hours).

The remaining of this section is structured as follows. We will first show an example of a set of solutions for one instance, then analyze each disruption independently and show the impact on all instances, and, finally, we will summarize all the results and look into the computation times.

Example. In Table 3 is an example of a set of solutions for Comp19 where one timeslot is made unavailable. It is seen that the minimum perturbation solution requires 7 perturbations and increases the solution value from 57 to 144 (153%) where the soft constraints are being more violated. By using one more perturbation, the quality can be much improved all the way down to 72, a 50% improvement. The additional perturbations gives smaller, but still significant improvements. The running times for each iteration are also shown, where it can be seen that the running times increase exponentially with the increased number of perturbations.

Objective	S_{Init}	S^{7}_{QR}	S^{8}_{QR}	S^{9}_{QR}	S^{10}_{QR}	S^{11}_{QR}	S^{12}_{QR}	S^{13}_{QR}	S^{14}_{QR}	S^{15}_{QR}
Time (s)	-	0	3	2	3	229	28	106	202	296
RoomCapacity	0	69	0	0	0	0	0	0	0	0
MinimumWorkingDays	5	15	10	5	5	5	5	5	5	5
CurriculumCompactness	52	56	56	56	54	54	52	52	52	52
RoomStability	0	4	6	6	6	5	6	5	5	4
f_{qual}	57	144	72	67	65	64	63	62	62	61

Table 3: Example with Comp19 with the *One Timeslot Unavailable* disruption, showing the objective values for each of the 9 different solutions. It takes 7 perturbations to reach a feasible solution with a quality that is 153% worse than the initial solution. However, by using an additional perturbation the quality get improved by 50%. Further perturbations gives more improvements.

4.1. One assignment invalid

The resulting full pareto fronts from making one assignment invalid on each of the 21 datasets is seen in Figure 2. Table 4 summarizes the results and shows the two lexicographic solutions, S^{min}_{QR} and S^{max}_{QR}. There is a significant difference between the data sets where the three instances comp01, comp02, and comp04 only need one perturbation to obtain a feasible solution with the same quality as the initial solution. The data set is different. Even though it can be made feasible using only one perturbation, it requires extra perturbations to recover as much quality as possible. However it is not able to obtain as high quality as the initial solution.

4.2. Insert curriculum

Adding a new curriculum of four courses impacts the timetable on each instance differently. Figure 3 shows the pareto fronts together with the value of the initial solution. comp01 and comp10 recover a lot of quality by allowing extra perturbations. A summary showing the two lexicographic solutions is seen in Table 5. instances can recover and regain the same value of f_{qual} as before the disruption.

4.3. Remove room whole day

The impact of removing a room for a whole day has a great impact on the solution, as shown in Figure 4. Especially comp20 loses a lot of quality, going from four to 148 penalty points. Table 6 summarizes the results showing that of the instances recovers the initial objective value.

Instance	S_{Init}	Disruption			Time (s)		$S_{\text{QR}}^{\text{min}}$		$S_{\text{QR}}^{\text{max}}$	
	f_{qual}	c	t	r	median	total	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	c0064	0,2	rS	0	0	1	†5	1	†5
comp02	24	c0313	4,4	r37	0	0	1	†24	1	†24
comp03	64	Mat1G2n	3,1	rG	8	163	1	65	4	†64
comp04	35	c1044	2,3	r52	0	0	1	†35	1	†35
comp05	285	IcoIcoB	5,2	r10	390	10,800	1	287	1	287
comp06	27	c0965	3,1	r36	318	10,800	1	30	2	28
comp07	6	c0007	4,2	r25	906	10,800	1	9	5	8
comp08	37	c0223	2,4	rG	409	10,800	1	42	7	40
comp09	96	c0535	1,0	r52	337	10,800	1	101	5	97
comp10	4	c0464	2,2	rDS1	246	10,800	1	10	6	9
comp11	0	c0027	2,6	rLUF2	0	0	1	10	2	†0
comp12	294	EtrAntIta	2,0	rO	173	10,800	1	313	5	307
comp13	59	c0163	2,1	rG	8	34	1	70	4	†59
comp14	51	c0184	4,0	rD	1	2	1	58	2	†51
comp15	62	ArcComCv	0,4	r27	304	10,800	1	70	9	67
comp16	18	c0199	0,2	rL	1	2	1	19	2	†18
comp17	56	c0143	2,1	rA	151	10,800	1	63	6	57
comp18	61	LetIta2	0,0	r1	266	10,800	1	76	11	64
comp19	57	c0036	4,3	r38	11	510	1	60	5	†57
comp20	4	c0537	1,0	r25	84	352	1	5	4	†4
comp21	74	c0474	1,3	r27	89	10,800	1	97	5	79

Table 4: Summary of the results of the *One assignment invalid* disruption. For each of the 21 instances, we show the initial solutions and the disruption column shows which combination of course, timeslot and room that was made unavailable by the disruption. We list the running time and the two lexicographic solutions i.e. the minimum perturbation solution and the best solution obtained by relaxing the perturbations. Ten instances recover completely and regain the same quality level as the initial solution, marked with †.

4.4. *One timeslot unavailable*

The final disruption we investigate is where an entire timeslot is made unavailable. Table 7 summarizes the results and shows that comp16 is infeasible and cannot be recovered. The other instances require between two and 17 perturbations to become feasible, which shows how large the impact of this disruption is. Figure 5 shows the full pareto fronts. Using more perturbations improves the quality a lot on all instances, but most of them are still far away from their initial solution value.

4.5. *Overall comparison*

As seen in the previous sections, the impact of the disruptions differs a lot between the disruption types and the data sets. In this section we will summarize the overall results for each disruption type.

Instance	S_{Init}	Disruption				Time (s)		$S_{\text{QR}}^{\text{min}}$		$S_{\text{QR}}^{\text{max}}$	
	f_{qual}	c_1	c_2	c_3	c_4	median	total	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	c0063	c0069	c0031	c0070	10	167	7	26	15	7
comp02	24	c0302	c0310	c0266	c0322	227	1,441	1	25	5	†24
comp03	64	StaAns	EsMn	Cos1Cv	GenAn	1	3	1	84	2	†64
comp04	35	c1007	c0420	c0527	c0670	3	33	3	40	6	†35
comp05	285	BibgraCS	Antrop	OriAnt	CatCla	713	10,800	2	291	9	287
comp06	27	c0184	c0959	c0484	c1058	1,184	10,800	3	43	9	30
comp07	6	c0069	c0072	c0897	c0489	213	10,800	1	13	2	8
comp08	37	c0179	c0441	c0978	c0420	529	10,800	2	40	6	38
comp09	96	c0117	c0107	c0112	c0503	206	10,800	2	110	8	98
comp10	4	c0443	c0963	c0069	c0515	376	10,800	4	26	13	6
comp11	0	c0028	c0109	c0107	c0036	0	0	2	†0	2	†0
comp12	294	CulM3	LatA	ItaCS	Geo1	1,000	10,800	2	347	9	299
comp13	59	c0249	c0218	c0036	c0506	4	8	1	60	2	†59
comp14	51	c1057	c0452	c0935	c0513	39	759	3	55	9	†51
comp15	62	Idr2Cn	AziGv	Mat1Cn	AppMv	372	10,800	5	74	13	70
comp16	18	c0184	c0143	c0965	c0442	95	10,800	3	30	8	23
comp17	56	c0128	c0220	c1031	c0600	6	12	2	57	3	†56
comp18	61	StoMed2	StoGre	Cod	Est	1	1	1	65	2	†61
comp19	57	c0035	c0055	c0511	c0114	278	10,800	4	70	12	60
comp20	4	c0526	c0467	c0201	c0455	1,776	10,800	3	22	5	21
comp21	74	c0439	c101e	c0463	c0261	3	10	3	119	5	†74

Table 5: The initial solution and the two lexicographic solutions for each instance with the *insert curriculum* disruption. instances the same solution value is completely recovered to the same value as the initial solution, marked with †.

Table 8 shows the average minimum number of perturbations across all datasets for each of the disruptions. It shows that the amount of perturbations needed to make the solution feasible differs a lot between the disruptions, from 1.0 to 11.3. This means that there is a wide variety of impact of the occurred disruptions on solutions.
Figure 6 shows that using additional perturbations decreases the number of violated soft constraints. For each dataset, the results are scaled so the x-axis is the number of extra perturbations relative to the minimum perturbation solution, meaning that zero is equal to the minimum perturbation solution i.e.

$$\bar{f}_\Delta = f_\Delta(S_{\text{QR}}^{\Delta}) - f_\Delta(S_{\text{QR}}^{\text{min}}) \tag{8}$$

The y-axis shows the decrease in quality relative to the initial solution, so 0% means that it reaches the same value, and 100% means that it is doubled, i.e.

$$\bar{f}_{qual} = 100 \cdot \left(\frac{f_{qual}(S_{\text{QR}}^{\Delta})}{f_{qual}(S_{\text{Init}})} - 1 \right) \tag{9}$$

Instance	S_{Init}	Disruption		Time (s)		$S_{\text{QR}}^{\text{min}}$		$S_{\text{QR}}^{\text{max}}$	
	f_{qual}	r	d	median	total	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	rS	3	12	309	5	16	14	6
comp02	24	rL	3	66	4,139	4	255	14	31
comp03	64	rE	3	12	1,044	4	72	10	†64
comp04	35	rF	4	84	3,871	2	37	8	†35
comp05	285	rB	5	120	4,161	2	287	8	†285
comp06	27	rN	4	725	10,800	2	29	2	29
comp07	6	r27	0	770	10,800	5	11	10	9
comp08	37	rD	1	299	10,800	4	57	13	40
comp09	96	r38	1	1	39	5	97	7	†96
comp10	4	rN	2	93	10,800	5	10	7	9
comp11	0	rLUF2	2	3	21	9	44	14	4
comp12	294	rL	2	410	10,800	4	312	9	300
comp13	59	rB	2	88	10,800	4	75	13	61
comp14	51	rN	1	115	1,347	4	54	9	†51
comp15	62	rDS1	0	178	10,800	4	66	11	63
comp16	18	r34	1	926	10,800	4	22	9	21
comp17	56	r27	0	818	10,800	5	59	5	59
comp18	61	rC1	0	8	106	2	63	6	†61
comp19	57	r37	0	311	6,029	2	59	9	†57
comp20	4	rF	3	41	10,800	5	148	10	15
comp21	74	r38	2	395	10,800	5	88	9	77

Table 6: For each instance we show the initial value and what day which room was removed by the *remove room whole day* disruption. We list the running times and the two lexicographic solutions. It can be seen that seven of the instances recovers to the same value as the initial solution.

To summarize on all the 21 instances we use quartiles due to the fact that the variation between the instances is high, and the distribution is skewed with a few very large values. The three lines indicate the 25%, 50% (median) and 75% quartile for the 21 instances.

Figure 6 shows that the median quality decrease for the minimum perturbation solution differs between 13% and up to 153% on the disruptions. It is also seen that the 25% most affected instances decreases in quality between 25% and 709%. The loss of quality is, therefore, significant in the minimum perturbation solution.

It is also seen that the quality increases significantly when introducing extra perturbations. For the median it is between , and on the 75% quartile the improvement is between . The distribution is skewed as the 25% and 50% quartiles are close, but the 75% is much further away. This shows that a few instances are heavily influences by the disruption compared to others.

Instance	S_{Init}	Disruption	Time (s)		$S_{\text{QR}}^{\text{min}}$		$S_{\text{QR}}^{\text{max}}$	
	f_{qual}	t	median	total	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	3,2	2	95	6	64	15	10
comp02	24	3,2	25	221	14	179	19	119
comp03	64	3,1	23	242	17	178	22	146
comp04	35	4,3	4	25	11	87	16	61
comp05	285	5,0	244	10,800	2	310	7	298
comp06	27	4,3	11	95	16	110	21	80
comp07	6	0,0	1,902	10,800	17	60	22	46
comp08	37	1,2	76	910	16	78	21	69
comp09	96	1,0	19	152	9	140	15	116
comp10	4	2,0	44	2,619	13	50	18	35
comp11	0	2,3	3	23	5	19	11	†0
comp12	294	2,3	44	2,082	10	463	15	383
comp13	59	2,1	11	88	15	143	20	123
comp14	51	1,0	11	356	10	137	15	65
comp15	62	0,4	293	2,032	12	105	17	93
comp16	18	1,0	0	0	-	-	-	-
comp17	56	0,4	198	3,679	11	96	16	86
comp18	61	0,0	434	10,800	3	84	11	68
comp19	57	0,0	28	870	7	144	15	61
comp20	4	3,0	56	2,353	17	159	22	71
comp21	74	2,3	98	1,296	15	190	20	160

Table 7: The summary of the *one timeslot unavailable* disruption. We list for each instance the initial value and the timeslot that was made unavailable. We then list the two lexicographic solutions. It is seen that comp16 is infeasible, and for the other instances, it takes between two and 17 perturbations to find a feasible solution.

Disruption	Avg. min. Perturbation (f_Δ)	Fully recovered
One Assignment Invalid	1.0	48%
Insert Curriculum	2.6	43%
Remove Room Whole Day	4.1	33%
One Timeslot Unavailable	11.3	5%
All	**4.8**	**32%**

Table 8: For each of the four disruptions, the average number of perturbations to make a solution feasible and the percentage of datasets that was able to recover to their initial solution quality after allowing more perturbations.

4.6. Computation time

In Table 9 is the average and median running times shown for each disruption. Across all disruptions is the average running time for an iteration 389 seconds but the median running time is seven seconds. This shows that a few iterations are responsible for most of the running time and that half of all iterations is solved in seven seconds or less. Figure 7 shows the running time for each iteration as a function of the number of perturbations for each of the four different disruptions. Overall is 38% of the iterations solved in less than 10 seconds. But as the number of perturbations increases so does the running time that can exceed two hours in a single iteration.

Solving static curriculum-based course timetabling problems to optimality is difficult, given it's NP-hardness Burke et al. (2010) but also in practice - as seen in Cacchiani et al. (2013) and Bettinelli et al. (2015). The included disruptions does not change the structure of the problem, therefore to fully recover a disrupted problem is still NP-hard. In this article we have applied Mixed Integer Programming models, solved by a standard solver, an approach which is not applicable to solve the problem from scratch. However, by adding limits on the allowed number of perturbations, we can apply the MIP solver with success. While we acknowledge that when this approach is implemented for a practical application, heuristics may be required, but we deemed it more important to obtain results with optimality bounds and possibly guarantees.

Disruption	Median time (s)	Avg. time (s)
One Assignment Invalid	94	1,033
Insert Curriculum	120	932
Remove Room Whole Day	56	880
One Timeslot Unavailable	21	361
All	**50**	**794**

Table 9: The average and median time for the iterations in each disruption and all. It shows that a few iterations are responsible for the majority of the total running time and that half of the iterations can be solved in 50 seconds or less.

5. Conclusion

In this paper, we have to recover feasibility of disrupted university timetables while taking quality of solutions into account. We have proposed a bi-objective optimization algorithm to find pareto-optimal solutions. This approach gives the planner the option to choose between several solutions and decide on the best trade-off between finding a timetable similar to the previous one and one with high quality.

Mixed Integer Programming is well suited for this task as it can prove infeasibility and find minimum perturbation solutions fast.

Future work. This is one of the very first studies that uses mixed integer programming for this problem. The approach is generic, and can be used on a broad variety of timetabling problems after formulating the static problem and the perturbation function using mixed-integer programming. The authors hope that this will spark further interest in the problem of recovering disrupted timetables. Both using this method on new problems and disruptions, but also investigate faster methods to solve this problem.

Note that Curriculum-based University course timetabling is NP-hard, as it can be reduced to a graph coloring problem (Burke et al., 2010). None of the disruptions imposed in this paper changes the overall structure of the problem, and therefore, to fully recover a disrupted problem to the optimal solution is NP-hard. Adding an upper bound on the number of perturbations limits the number of feasible solutions, but the number increases exponentially depended on the bound. Better performance could therefore maybe be expected by using dedicated methods that have proven well on timetabling problems, and heuristics could also be a very powerful practical method, though loosing the optimality criteria.

Acknowledgement

We would like to thank the organizers of the Second International Timetabling Competition for formulating the Curriculum-Based Course Timetabling problem. We would also like to thank Alex Bonutti, Luca Di Gaspero and Andrea Schaerf for creating and maintaining the website with instances and best-known solutions.

References

Barták, R., Müller, T., Rudová, H., 2003. A new approach to modeling and solving minimal perturbation problems. In: International Workshop on Constraint Solving and Constraint Logic Programming. Springer, pp. 233–249.

Bettinelli, A., Cacchiani, V., Roberti, R., Toth, P., 2015. An overview of curriculum-based course timetabling. TOP, 1–37.

Bonutti, A., De Cesco, F., Di Gaspero, L., Schaerf, A., April 2012. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. Annals of Operations Research 194 (1), 59–70.

Burke, E. K., Mareček, J., Parkes, A. J., Rudová, H., 2008. Penalising patterns in timetables: Novel integer programming formulations. In: Kalcsics, J., Nickel, S. (Eds.), Operations Research Proceedings 2007. Vol. 2007 of Operations Research Proceedings. Springer Berlin Heidelberg, pp. 409–414, 10.1007/978-3-540-77903-2_63.

Burke, E. K., Mareček, J., Parkes, A. J., Rudová, H., 2010. A supernodal formulation of vertex colouring with application in course timetabling. Annals of Operations Research 179 (1), 105–130.

Cacchiani, V., Caprara, A., Roberti, R., Toth, P., 2013. A new lower bound for curriculum-based course timetabling. Computers & Operations Research 40 (10), 2466 – 2477.

Di Gaspero, L., McCollum, B., Schaerf, A., 2007. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Tech. rep., School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom.

Elkhyari, A., Gueret, C., Jussien, N., 2003. Solving dynamic resource constraint project scheduling problems using new constraint programming tools. In: Burke, E., De Causmaecker, P. (Eds.), Practice and Theory of Automated Timetabling IV. Vol. 2740 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 39–59.

Ferland, J. A., Roy, S., 1985. Timetabling problem for university as assignment of activities to resources. Computers & operations research 12 (2), 207–218.

Fischetti, M., Lodi, A., 2003. Local branching. Mathematical Programming 98, 23–47.

Fukunaga, A., 2013. An improved search algorithm for min-perturbation. In: International Conference on Principles and Practice of Constraint Programming. Springer, pp. 331–339.

Haimes, Y. Y., Ladson, L., Wismer, D. A., 1971. Bicriterion formulation of problems of integrated system identification and system optimization.

Kocjan, W., 2002. Dynamic scheduling. state of the art report. SICS Research Report.

Lach, G., Lübbecke, M., 2008. Optimal university course timetables and the partial transversal polytope. In: McGeoch, C. (Ed.), Experimental Algorithms. Vol. 5038 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 235–248.

Müller, T., Rudová, H., Barták, R., 2005. Minimal perturbation problem in course timetabling. In: Burke, E., Trick, M. (Eds.), Practice and Theory of Automated Timetabling V. Vol. 3616 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 126–146.

Phillips, A. E., Walker, C. G., Ehrgott, M., Ryan, D. M., 2016. Integer programming for minimal perturbation problems in university course timetabling. Annals of Operations Research, 1–22.

Phillips, A. E., Waterer, H., Ehrgott, M., Ryan, D. M., 2015. Integer programming methods for large-scale practical classroom assignment problems. Computers & Operations Research 53 (0), 42 – 53.
URL `http://www.sciencedirect.com/science/article/pii/S0305054814001956`

Rudová, H., Muller, T., Murray, K., 2011. Complex university course timetabling. Journal of Scheduling 14 (2), 187–207.

Sakkout, H., Wallace, M., 2000. Probe backtrack search for minimal perturbation in dynamic scheduling. Constraints 5 (4), 359–388.
URL `http://dx.doi.org/10.1023/A\%3A1009856210543`

Verfaillie, G., Jussien, N., 2005. Constraint solving in uncertain and dynamic environments: A survey. Constraints 10 (3), 253–281.

Zivan, R., Grubshtein, A., Meisels, A., 2011. Hybrid search for minimal perturbation in dynamic csps. Constraints 16 (3), 228–249.

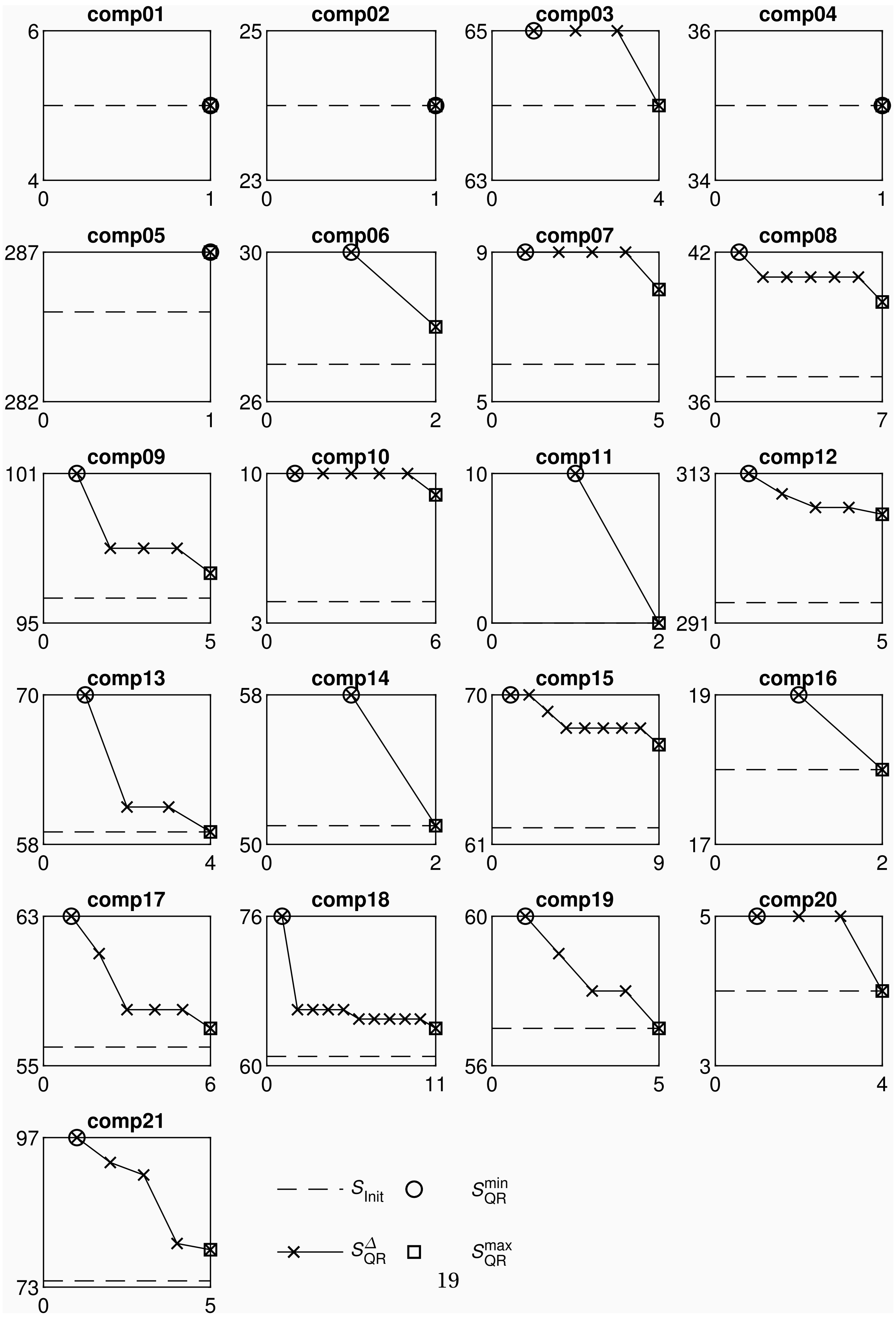

Figure 2: The pareto fronts for each data set on the *one assignment invalid* disruption and the objective value of the initial solution. The x-axis is f_Δ and the y-axis is f_{qual}. There is a large difference between how much worse the quality is and how much is gained by using extra perturbations.

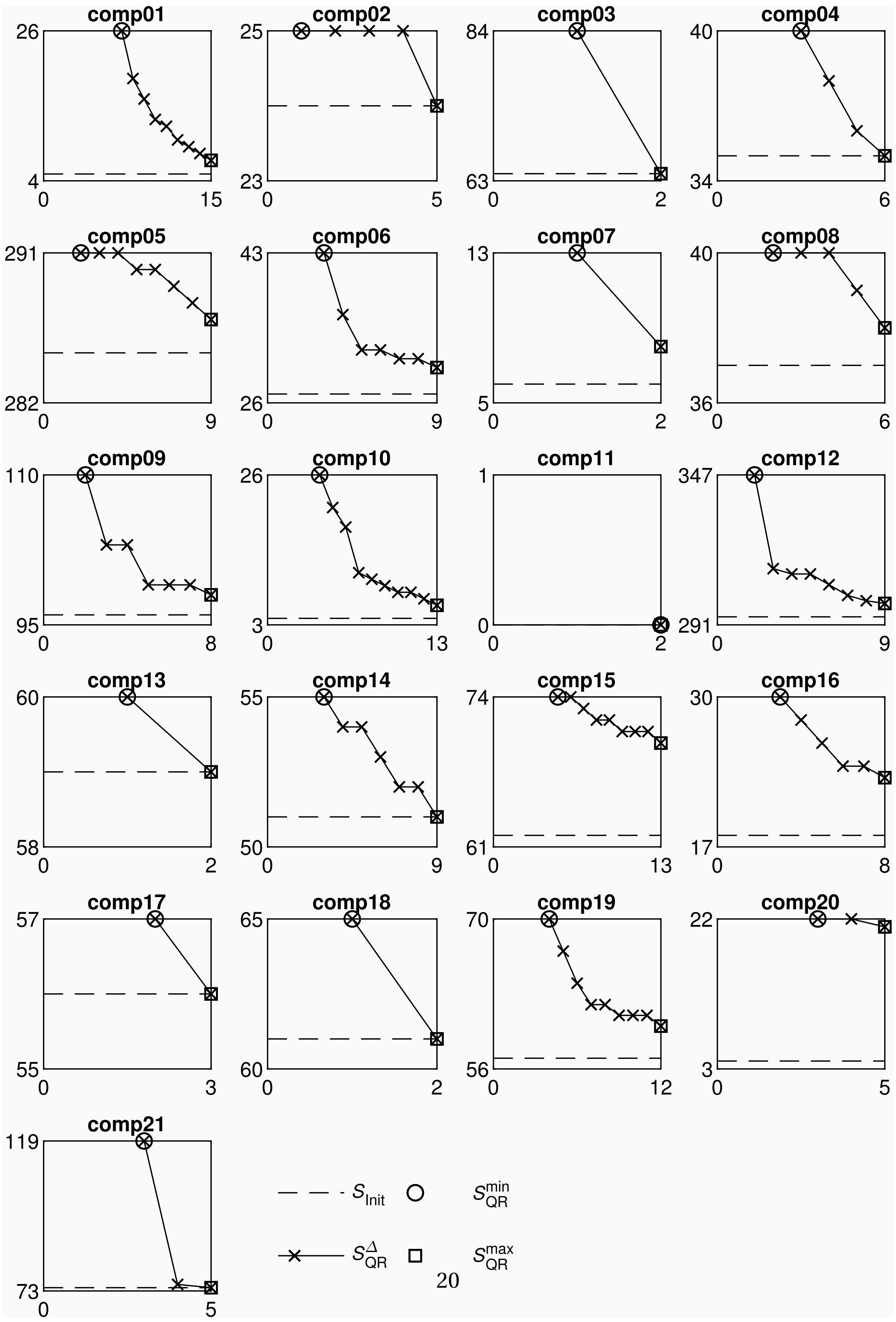

Figure 3: The pareto fronts for the *insert curriculum* disruption. The x-axis is f_{Δ} and the y-axis is f_{qual}. Especially the two instances comp01 and comp10 recover a lot of quality from extra perturbations.

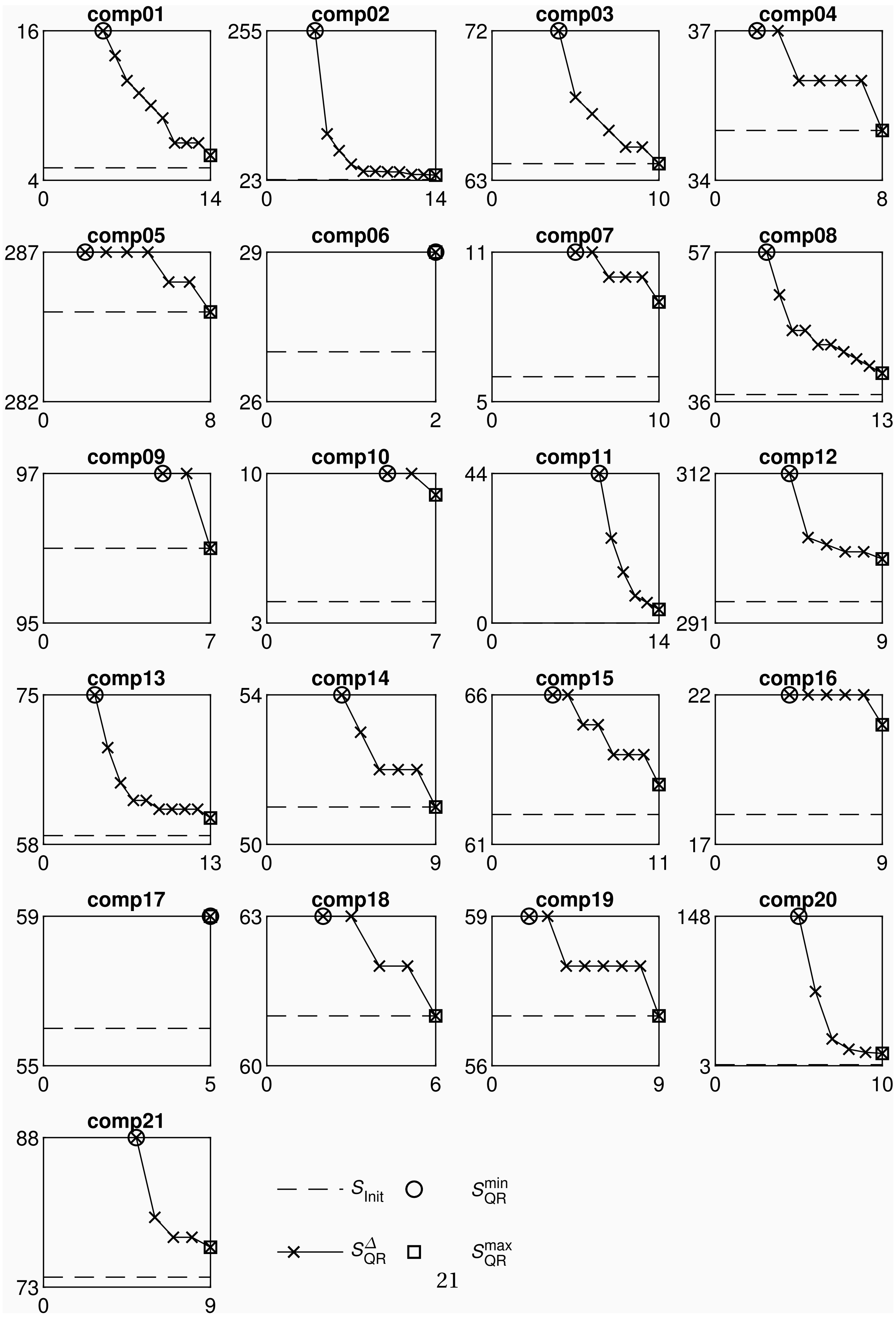

Figure 4: The pareto front for the *remove room whole day* disruption. The x-axis is f_{Δ} and the y-axis is f_{qual}. Especially comp20 is affected and uses many extra perturbations to recover most of the quality.

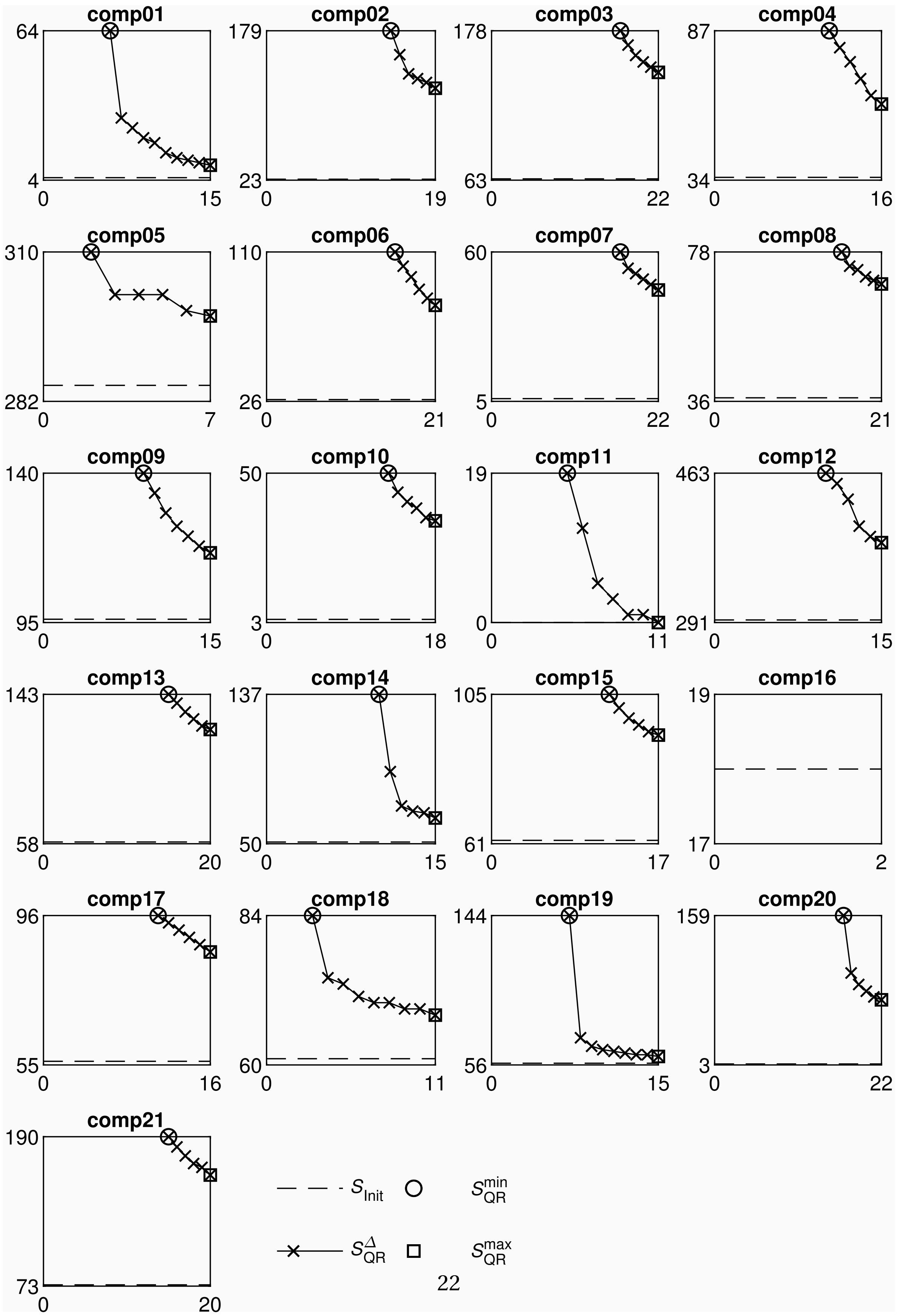

Figure 5: The pareto fronts for the *one timeslot unavailable* disruption. The x-axis is f_Δ and the y-axis is f_{qual}. The impact is high on all instances and comp16 is infeasible and cannot be recovered.

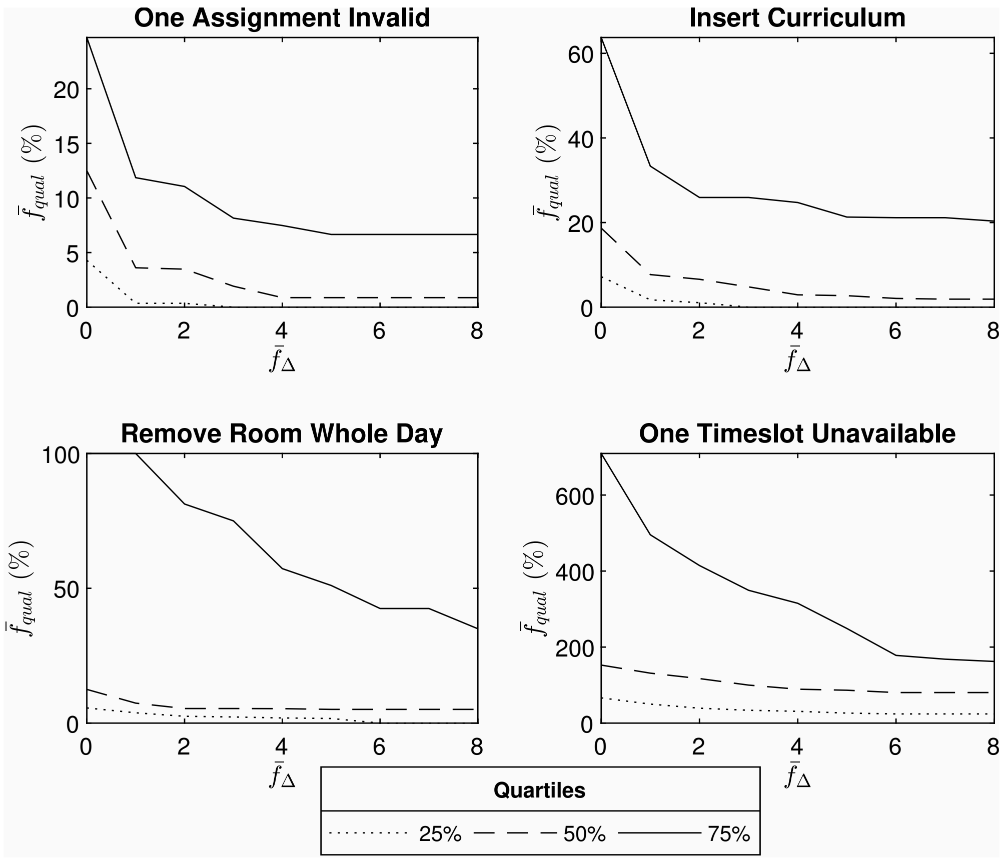

Figure 6: The quality decrease in the minimum perturbation solution and how it increases by using extra perturbations for each of the different disruptions. The three lines show the quartiles that indicate that there is a significant variance between the datasets.

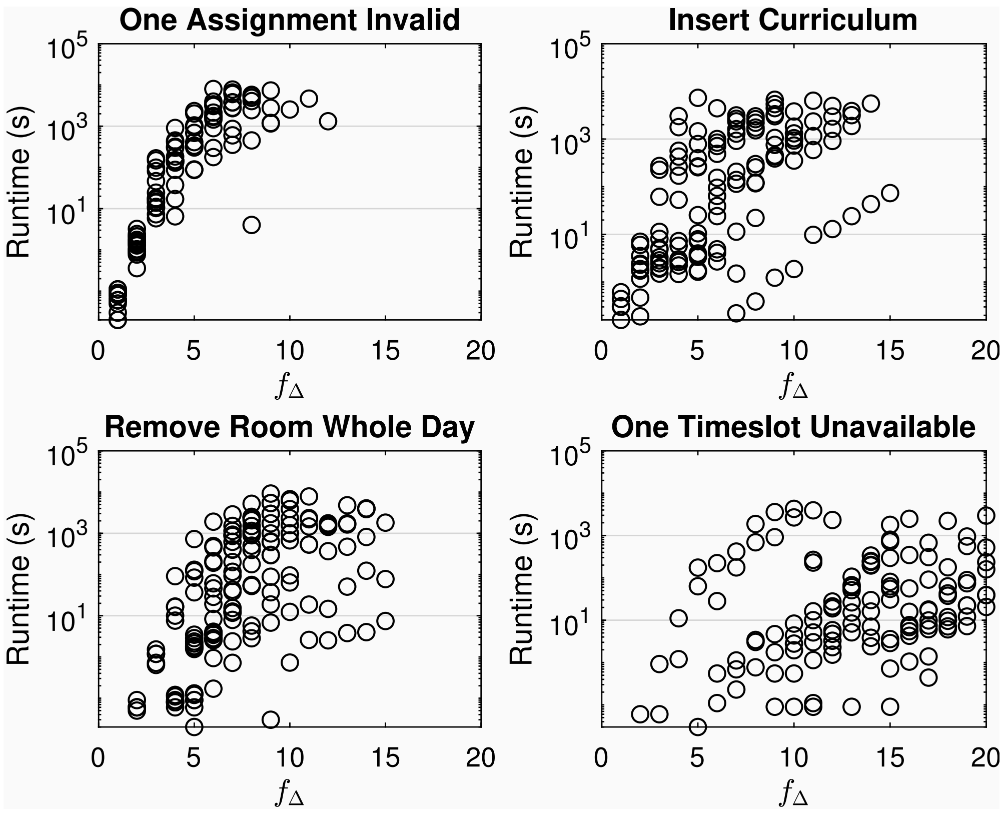

Figure 7: Running times for each iteration as a function of the number of perturbations for each of the four different disruptions. In total of the models are solved in less than 10 seconds. But when the number of perturbations increases the solution time can get up to hours.