Please update a copy of
scan-of-declaration-of-authorship.pdf

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED

Bachelor Thesis

Ein systematischer Vergleich von Ansätzen zum "Student Course Timetable Problem" am Beispiel der Semesterplanung Studierender der Wirtschaftsuniversität Wien im Bachelorstudium

Philipp Scheer

Date of Birth: 28.01.2003
Student ID: 12242922

Subject Area: Information Business

Studienkennzahl: J123456789

Supervisor: Assoz.Prof PD Dr. Stefan Sobernig

Date of Submission: XX. XXX 2025

Department of Information Systems & Operations Management, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR WIRTSCHAFTSINFORMATIK UND OPERATIONS MANAGEMENT DEPARTMENT OF INFORMATION SYSTEMS AND OPERATIONS MANAGEMENT

Contents

List of Figures

List of Tables

Abstract

Todo

1 Einleitung

Kaum eine Entscheidung im Alltag von Studierenden erscheint so simpel und erweist sich gleichzeitig so komplex wie die Erstellung des eigenen Stundenplans. Was auf den ersten Blick nach einer organisatorischen Routineaufgabe aussieht, entpuppt sich bei näherer Betrachtung als vielschichtiges Optimierungsproblem: Überschneidungen zwischen Lehrveranstaltungen, begrenzte Kursplätze und individuelle Präferenzen wie Arbeitszeiten und Lerngewohnheiten machen die Planung zu einer Herausforderung

In einer Zeit, in der viele Entscheidungsprozesse bereits durch Algorithmen unterstützt werden, stellt sich die Frage, ob auch die Stundenplanerstellung automatisiert und optimiert werden kann. Hier setzt die folgende Arbeit an: Sie untersucht, wie sich das *Student Schedulung Problem (SSP)* mithilfe von verschiedenen Optimierungsverfahren modellieren und lösen lässt.

2 Hintergrund

2.1 Motivation

Studierende sehen sich im Studium mit einem umfangreichen Kursangebot konfrontiert und müssen ihre Stundenpläne meist händisch zusammenstellen. Dieser manuelle Prozess ist jedoch mehr als eine rein administrative Aufgabe; er ist eine Optimierungsentscheidung, bei der persönliche und akademische Rahmenbedingungen gegeneinander abgewogen werden müssen.

Studierende berichten häufig von Schwierigkeiten bei der Erstellung ihrer Semesterpläne [6, 11, 13]. Zu den zentralen Ursachen zählt insbesondere die Vereinbarkeit mit einem Nebenjob; so bevorzugen 81% der Befragten mindestens einen vollständig freien Wochentag [13]. Zudem geben 74% der Studierenden an, dass eine gleichmäßige Verteilung von Lehrveranstaltungen und Prüfungen für sie das wichtigste Kriterium bei der Stundenplanerstellung darstellt [7]. Weiterhin möchten 82% vermeiden, mehr als eine Prüfung pro Tag zu absolvieren [7]. Bezüglich der zeitlichen Präferenzen werden Mittagsstunden am häufigsten gewählt, gefolgt von frühen und späten Zeitslots. Rund 31% der Studierenden bevorzugen es, an bestimmten Tagen keine Prüfungen zu haben; besonders unpopulär sind Samstag, Sonntag, Freitag und Montag [7].

Obwohl sich diese Präferenzen grundsätzlich leicht modellieren lassen, stehen derzeit keine Programme zur Verfügung, die eine automatisierte Erstellung entsprechender Stundenpläne an der WU ermöglichen. Lediglich der Studienplaner der Österreichischen Hochschüler*innenschaft der WU [20] erlaubt die manuelle Kombination von Lehrveranstaltungen und Prüfungen im Hinblick auf Überschneidungsfreiheit, bietet jedoch keine Möglichkeit zur Optimierung nach individuellen Präferenzen.

Diese Diskrepanz zwischen den Anforderungen der Studierenden und den limitierten bestehenden Programmen führt zu einer suboptimalen Planungssituation.

2.2 Problemstellung

Die Erstellung eines optimalen Semesterplans ist eine zentrale Herausforderung für Studierende. Das Angebot an Lehrveranstaltungen lässt sich nicht frei kombinieren, da auf zeitliche Konflikte, Erfüllung einer Mindestanzahl an ECTS pro Semester, Arbeitstätigkeit, etc. geachtet werden muss. Die Herausforderung besteht darin, aus diesem Pool an Lehrveranstaltungen eine zulässige und optimale Kombination zu wählen. Zulässig gilt eine Zeittafel, wenn keine zeitlichen Konflikte auftreten (keine *Hard Constraints*

wurden verletzt). Optimal ist eine Zeittafel, wenn eine gegebene Zielfunktion maximal ist. Die Zielfunktion beinhaltet alle Präferenzen des Studierenden, die seine Lebensrealtät widerspiegeln sollen (*Soft Constraints*). Dazu zählen etwa die Berücksichtigung von Arbeitstätigkeiten, die durch Blockieren oder Priorisieren von gewissen Stunden in der Woche ausgedrückt werden, sowie die Favorisierung von Kursen, die durch eine Priorität ausgedrückt wird. Diese Problemstellung wird in der Literatur als Zeittafelproblem oder *Student Scheduling Problem (SSP)* bezeichnet.

2.3 Zeittafelprobleme

Die Erstellung von Zeitplänen im akademischen Kontext gliedert sich in eine Hierarchie jeweils eigenständiger Optimierungsaufgaben. Diese Hierarchie reicht von der strategischen Planung der gesamten Universität bis hin zur individuellen Semesterplanung des einzelnen Studierenden. Um diese Arbeit einzuordnen, ist es notwendig, die theoretische Abstufung vom University Course Timetabling über das Student Sectioning bis hin zum Single Student Scheduling zu verstehen.

2.3.1 University Course Timetabling Problem (UCTTP)

Auf der obersten Ebene steht das University Course Timetabling Problem (UCTTP). Dieses Problem befasst sich mit der Erstellung des "Master-Stundenplans". Die Kernaufgabe besteht darin, eine Menge von Lehrveranstaltungen (Events) einer Menge von Zeitfenstern und Räumen zuzuordnen [15]. Dabei müssen Ressourcenbeschränkungen (z. B. Raumgrößen) und *Hard Constraints*, wie die Verfügbarkeit von Dozenten, berücksichtigt werden.

Carter und Laporte beschreiben dieses Problem als NP-hart [5]. Das bedeutet, dass mit steigender Anzahl an Veranstaltungen und Restriktionen der Rechenaufwand für eine exakte Lösung exponentiell wächst. Das Ziel des UCTTP ist hauptsächlich die Machbarkeit und Effizienz aus Sicht der Institution: Es muss sichergestellt werden, dass alle Kurse stattfinden können, ohne dass Dozenten oder Räume doppelt belegt werden. Individuelle Studierendenpräferenzen spielen auf dieser Ebene oft nur eine untergeordnete Rolle oder werden aggregiert betrachtet.

2.3.2 Student Sectioning Problem (SSP)

Sobald der Master-Stundenplan fixiert ist (die Veranstaltungen haben feste Zeiten und Räume), folgt die nächste Ebene: das Student Sectioning Problem

(SSP). Hier verlagert sich der Fokus von der Ressourcenzuordnung (Wann findet der Kurs statt?) zur Belegung (Wer besucht welchen Kurs?).

Das SSP befasst sich mit der Verteilung der Studierenden auf Übungsgruppen oder Parallelveranstaltungen (Sections) eines Kurses [12]. Ein klassisches Beispiel ist ein Modul, das aus einer zentralen Vorlesung und zehn verschiedenen Labor-Terminen besteht. Das Ziel ist es, alle Studierenden so auf die Labore zu verteilen, dass Überschneidungen vermieden und Kapazitätsgrenzen eingehalten werden. Müller et al. unterscheiden hierbei zwei Ansätze [12]:

1. **Batch Sectioning:** Alle Studierendenwünsche werden gesammelt und global optimiert. Das maximiert die Fairness und die Gesamtauslastung, findet aber meist vor Semesterbeginn statt.

2. **Online Sectioning:** Die Zuteilung erfolgt sequenziell in Echtzeit (z. B. während der Anmeldephase via LPIS im Kontext der WU). Dies entspricht einem "Greedy"-Ansatz, bei dem für späte Anmeldungen oft nur suboptimale Restplätze vergeben werden können.

2.3.3 Single Student Scheduling

Die granularste Ebene der Hierarchie — und der Fokus dieser Bachelorarbeit — ist das Single Student Scheduling. Während das SSP versucht, eine Menge von Studierenden global auf Kurse aufzuteilen, betrachtet das Single Student Scheduling das Problem ausschließlich aus der Perspektive eines einzelnen Studierenden.

Die Fragestellung lautet hier nicht: "Wie fülle ich die Kursräume optimal?", sondern: "Welche Kombination aus den verfügbaren Kursangeboten maximiert meinen persönlichen Nutzen?" [8].

Obwohl der Suchraum für einen einzelnen Studierenden deutlich kleiner ist als für die gesamte Universität, bleibt die Komplexität hoch. Da ein Studierender oft aus hunderten möglichen Kombinationen von Vorlesungen und Übungen wählen kann, die unterschiedliche zeitliche Attribute und Abhängigkeiten aufweisen, handelt es sich auch hier um ein kombinatorisches Optimierungsproblem. Die Herausforderung besteht darin, harte Restriktionen (keine zeitlichen Überschneidungen) zu erfüllen und gleichzeitig weiche Restriktionen (z. B. "keine Kurse am Freitag", "kompakter Stundenplan") zu optimieren.

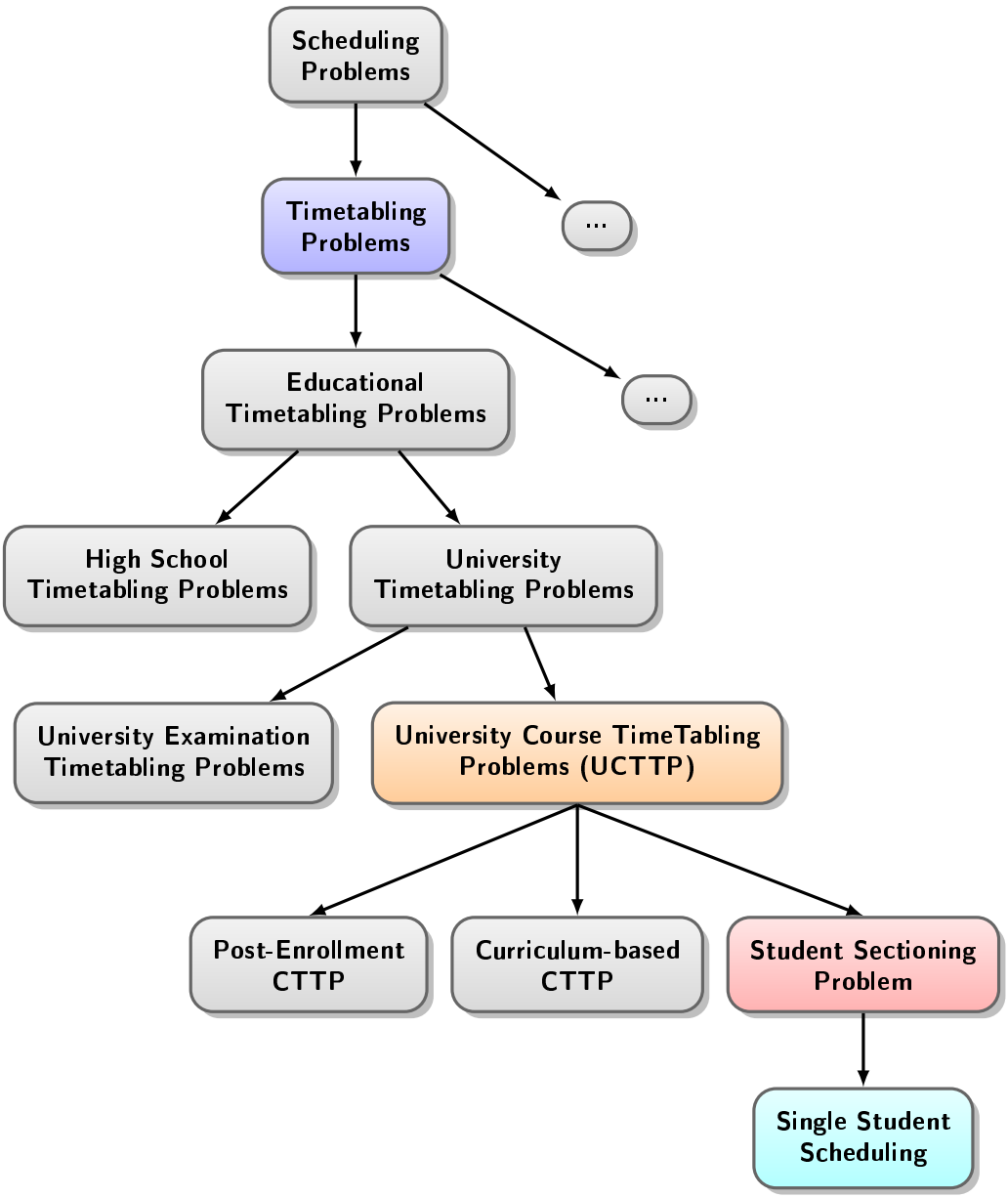


Figure 1: Hierarchie von Zeittafelproblemen (Adaptiert aus [10])

2.4 Replikationsarbeit

Die Replizierbarkeit von Forschungsergebnissen gilt als grundlegender Eckpfeiler der Wissenschaft und ist essenziell für den Aufbau eines kumulativen Wissensbestandes. In der wissenschaftlichen Praxis wird unter einer Replikation im Allgemeinen der Versuch verstanden, die Ergebnisse einer früheren Studie durch die erneute Durchführung derselben oder einer ähnlichen Untersuchung zu bestätigen oder zu widerlegen. Das Ziel ist es, die Verlässlichkeit (Reliabilität) und Gültigkeit (Validität) von wissenschaftlichen Erkenntnissen zu prüfen und sicherzustellen, dass veröffentlichte Effekte nicht auf Zufall, Fehlern oder spezifischen Artefakten der ursprünglichen Studie beruhen.

2.4.1 Definition und Arten der Replikation

In der Literatur wird häufig zwischen verschiedenen Arten der Replikation unterschieden, wobei die Terminologie je nach Disziplin variiert. Ein zentraler Unterschied besteht zwischen der "exakten" oder "nahen" Replikation (Close Replication) und der "konzeptionellen" Replikation.

- Bei der **exakten Replikation** wird versucht, die Methoden und Prozeduren der Originalstudie so exakt wie möglich zu reproduzieren. Das Ziel ist es, idealerweise nur jene Unterschiede zuzulassen, die unvermeidbar sind (z. B. ein anderer Zeitpunkt oder eine andere Stichprobe aus derselben Population).

- **Konzeptionelle Replikationen und Erweiterungen** überprüfen die Robustheit der Ergebnisse unter veränderten Rahmenbedingungen, beispielsweise durch die Verwendung neuer Datensätze, variierter Parameter oder alternativer Schätzverfahren. Bonett [2] bezeichnet dies auch als *comparable follow-up study*, bei der zwar das gleiche Design und die gleichen Messskalen verwendet werden, aber die Population oder der Kontext variieren kann.

Für die computergestützte Forschung (Computational Research) definiert Sandve et al. [14] Reproduzierbarkeit zudem als die Fähigkeit, Ergebnisse durch eine transparente Dokumentation des gesamten Analyse-Workflows — von den Rohdaten bis zum Endergebnis — nachvollziehbar und wiederholbar zu machen.

2.4.2 Anforderungen an eine wissenschaftliche Replikation

Damit eine Replikationsarbeit wissenschaftlichen Standards genügt, müssen bestimmte Anforderungen erfüllt sein. Brandt et al. [3] entwickelten hierfür das *Replication Recipe*, welches unter anderem folgende Kriterien hervorhebt:

- **Exakte Definition der zu replizierenden Methoden:** Die Methoden der Originalstudie müssen sorgfältig analysiert und so getreu wie möglich nachgebildet werden.
- **Hohe statistische Aussagekraft:** Die Replikationsstudie muss über eine ausreichende Stichprobengröße verfügen, um Effekte zuverlässig nachweisen zu können.
- **Transparenz und vollständige Dokumentation:** Alle Details zur Durchführung, inklusive Code und Daten, sollten verfügbar gemacht werden, um eine Überprüfung durch Dritte zu ermöglichen.
- **Kritischer Vergleich:** Die Ergebnisse der Replikation müssen statistisch fundiert mit denen der Originalstudie verglichen werden, anstatt sich rein auf Signifikanztests zu verlassen.

Im Bereich der Management-Mathematik und Optimierung wird außerdem gefordert, dass Algorithmen so spezifiziert sind, dass unabhängige Forscher dieselben Ergebnisse erzielen können. Burman et al. [4] betonen weiters die Notwendigkeit, dass der replizierende Forscher unabhängig von den ursprünglichen Autoren ist.

2.4.3 Einordnung und Umsetzung in dieser Arbeit

Die vorliegende Bachelorarbeit ordnet sich als Replikationsstudie der Arbeit von Feldman und Golumbic [9] ein. Sie kann als konzeptionelle Replikation mit Erweiterung klassifiziert werden.

Ziel ist es nicht, die exakten Ergebnisse von 1990 mit den damaligen Daten zu reproduzieren, sondern die Validität der vorgeschlagenen Lösungsansätze (Hill Climbing und Offering Order) in einem neuen Kontext zu überprüfen. Dies entspricht der von Burman et al. [4] beschriebenen "validierenden Replikation", bei der geprüft wird, ob Ergebnisse robust gegenüber Veränderungen in Datensätzen und Zeiträumen sind.

Um die wissenschaftlichen Anforderungen an diese Replikation zu erfüllen, werden in dieser Arbeit folgende Maßnahmen getroffen:

- **Methodische Treue:** Die Algorithmen werden basierend auf der Beschreibung in der Originalarbeit von Feldman und Golumbic [9] exakt nachprogrammiert (*re-implemented*), um die methodische Vergleichbarkeit zu gewährleisten.

- **Neuer Datensatz:** Die Evaluation erfolgt anhand eines aktuellen, realen Datensatzes der Wirtschaftsuniversität Wien (Sommersemester 2025), um die Anwendbarkeit auf moderne Student Scheduling Problems (SSP) zu testen.

- **Erweiterte Baseline:** Da der ursprüngliche Vergleichsmaßstab (Brute Force) für die Komplexität des neuen Datensatzes nicht praktikabel ist, wird eine neue Baseline mittels Integer Linear Programming (ILP) eingeführt. Dies stellt eine methodische Modernisierung dar, wahrt aber das Ziel, die heuristischen Verfahren gegen eine optimale Lösung zu benchmarken.

- **Transparenz (Computational Reproducibility):** Im Sinne der "Ten Simple Rules" von Sandve et al. [14] werden alle Schritte der Datenextraktion und Analyse dokumentiert. Der Quellcode der Implementierungen ist, wie in der Arbeit referenziert, öffentlich einsehbar, um vollständige Nachvollziehbarkeit zu garantieren.

Durch dieses Vorgehen leistet die Arbeit einen Beitrag zur Überprüfung der Generalisierbarkeit historischer Optimierungsansätze unter modernen Rahmenbedingungen.

2.5 Zielsetzung und Abgrenzung

Die vorliegende Bachelorarbeit hat zum Ziel, die beschriebene Problemstellung unter den aktuellen technischen Rahmenbedingungen mithilfe verschiedener Lösungsansätze zu untersuchen, diese anhand eines aktuellen Datensatzes zu validieren und anschließend einem Leistungsvergleich zu unterziehen.

1. **Validierung:** Im Mittelpunkt steht die exakte Nachprogrammierung der Algorithmen von Feldman und Golumbic [9]. Es soll überprüft werden, ob diese Verfahren unter modernen Rechenbedingungen und anhand realer Daten aus dem Vorlesungsverzeichnis der Wirtschaftsuniversität Wien in der Lage sind, einen optimalen Semesterplan zu erstellen.

2. **Vergleichende Leistungsanalyse:** Die implementierten Algorithmen werden hinsichtlich der Qualität der erreichten Lösungen (*Score*), ihrer Geschwindigkeit sowie ihres Speicherbedarfs systematisch miteinander verglichen.

Folgende Aspekte sind explizit vom Forschungsumfang ausgeschlossen:

1. Das Problem wird ausschließlich als Single Student Scheduling für einen einzelnen Studierenden gelöst. Es wird **keine Rücksicht auf andere Studierende**, die maximale Kapazität von Räumen, die Verfügbarkeit von Professoren oder die allgemeine universitäre Ressourcenplanung genommen.

2. Die Planung beschränkt sich auf **ein einzelnes Semester** des Hauptstudium Wirtschaftsinformatik.

3. Die Entwicklung einer **voll funktionsfähigen Applikation** zur Dateneingabe, Speicherung von Planungen, oder komplexer Visualisierung wird ausgeschlossen. Das Ergebnis des Programms wird durch ein computerlesbares Format (JSON) ausgegeben.

3 Datenbereitstellung

Um einen möglichst aktuellen Datensatz zu erzeugen, wurden die Vorlesungsdaten aus dem Vorlesungsverzeichnis der Wirtschaftsuniversität Wien [1] automatisiert ausgelesen. Jedem Kurs (*Offering*) ist eine Lehrveranstaltungs-ID zugewiesen, welche durch eine Zahl von 1 bis 9999 dargestellt wird. Weiters gehört jeder Kurs einem Planpunkt (*Course*) an.

3.1 Struktur des Datensatzes

Der Datensatz des Vorlesungsverzeichnisses der Wirtschaftsuniversität Wien für das Sommersemester 2025 bildet die Grundlage für die vorliegende Arbeit. Die Daten wurden automatisiert unter Verwendung des beschriebenen Prozesses 1 extrahiert.

Die Modellierung des Studiums an der WU basiert auf zwei zentralen Entitäten, die für das Verständnis der Optimierungsalgorithmen wichtig sind:

- **Course (Planpunkt):** Ein *Course* repräsentiert ein zu absolvierendes Modul, das mit einer festen ECTS-Anzahl assoziiert ist. Aus den verfügbaren *Offerings* eines Planpunkts darf maximal eines ausgewählt werden.
- **Offering (Lehrveranstaltung):** Ein *Offering* ist eine Lehrveranstaltung, die einem Planpunkt angehört. Jedes Offering besitzt eine eindeutige ID (von $1 - 9999$) und ist mit konkreten Termin- und Zeitangaben hinterlegt, die für die Prüfung von zeitlichen Konflikten verwendet wird.

3.2 Extraktionsprozess

Der Prozess der automatisierten Extraktion wurde folgendermaßen realisiert:

Algorithm 1 Datenextraktion aus Vorlesungsverzeichnis

1: $offerings \leftarrow \emptyset$
2: $modules \leftarrow \emptyset$
3: **for** $id \leftarrow 1, 9999$ **do**
4: $\quad url \leftarrow$ CraftCourseURL(id)
5: $\quad page \leftarrow$ HttpGet(url)
6: $\quad$ **if** Contains($page$"$Keine\ Lehrveranstaltung\ gefunden$") **then**
7: $\quad\quad$ **continue**
8: $\quad$ **end if**
9: $\quad (module, dates) \leftarrow$ ExtractDates($page$)
10: $\quad$ **if** $module \notin \mathrm{dom}(modules)$ **then**
11: $\quad\quad ects \leftarrow$ ExtractECTSFromModule($module$)
12: $\quad\quad modules \leftarrow modules \cup \{(module, ects)\}$
13: $\quad$ **end if**
14: $\quad offerings \leftarrow offerings \cup \{(id, dates, modules_{module})\}$
15: **end for**

Das Ergebnis sieht wie folgt aus:

offeringId	dates	lvLeiter	module_ids	ects
5877	[{"start": "2025-03-11 09:00", "end": "2025-03-11 11:00"}, ...]	Assoz.Prof PD Dr. Sabrina Kirrane	["6590", "9485"]	5
6427	[{"start": "2025-03-13 16:30", "end": "2025-03-13 20:30"}, ...]	Dr. Robert Kosik	['5160']	6
4676	[{"start": "2025-05-06 14:30", "end": "2025-05-06 17:00"}, ...]	Dr. Stefan Treitl	['6012']	4
6295	[{"start": "2025-03-06 08:00", "end": "2025-03-06 12:00"}, ...]	Anna-Lena Klug, MSc. MSc.	['5155']	8
5745	[{"start": "2025-03-10 18:00", "end": "2025-03-10 20:00"}, ...]	Dr. Reinhard Zuba	['5108']	4
6258	[{"start": "2025-04-01 14:00", "end": "2025-04-01 19:00"}, ...]	Dr. Nikolaus Obwegeser	['5162']	3
5944	[{"start": "2025-05-08 09:30", "end": "2025-05-08 12:00"}, ...]	Univ.Prof. Dr. Jonas Puck	['5107']	4
5724	[{"start": "2025-03-10 16:00", "end": "2025-03-10 20:00"}, ...]	Alexander Oberreiter, M.Sc.	['5105']	8
5752	[{"start": "2025-05-06 12:00", "end": "2025-05-06 14:30"}, ...]	Univ.Prof. Dipl.Wirtsch.-Math.Dr. Birgit Rudloff	['6023']	4
6354	[{"start": "2025-05-06 15:30", "end": "2025-05-06 18:00"}, ...]	Zhenyi Wang, M.Sc.	['5059', '6059']	4
6447	[{"start": "2025-03-13 12:30", "end": "2025-03-13 16:30"}, ...]	Anita Neumannova, PhD, MSc (WU), MIM (CEMS)	['5161']	4
5329	[{"start": "2025-09-01 14:00", "end": "2025-09-01 17:00"}, ...]	Dr. Tingmingke Lu	['5056', '6056']	4
6317	[{"start": "2025-03-05 14:00", "end": "2025-03-05 18:00"}, ...]	Dipl.-Wirt.-Ing.(FH) Mark Nicolas Grünsteidl	['5158']	8
6130	[{"start": "2025-05-06 09:30", "end": "2025-05-06 12:00"}, ...]	Jana Hlavinova, Ph.D.	['6024']	4
6182	[{"start": "2025-03-11 10:00", "end": "2025-03-11 12:00"}, ...]	Dr. Sonja Sperber	['5136', '6911']	3
6089	[{"start": "2025-09-10 09:00", "end": "2025-09-10 13:00"}, ...]	Dr. Nikolai Neumayer	['5106']	4
4086	[{"start": "2025-05-06 14:30", "end": "2025-05-06 17:00"}, ...]	Vanessa Kofler, LL.M. (WU)	['5109', '6021']	4
5762	[{"start": "2025-03-10 16:30", "end": "2025-03-10 18:00"}, ...]	Univ.Prof. Jonas Bunte, Ph.D.	['5117']	4

Table 1: Auszug des Ergebnisses der Datenbereitstellung

3.3 Deskriptive Analyse

Um die Komplexität des Datensatzes und des Optimierungsproblems zu quantifizieren, wurde eine deskriptive Analyse durchgeführt.

Der vollständige Datensatz besteht aus insgesamt 381 *Offerings* ($|\mathcal{O}|$) und 28 *Courses*, woraus sich eine durchschnittliche Anzahl von 13.6 *Offerings* pro *Course* ergibt.

Ein wesentlicher Faktor für die Schwierigkeit der Stundenplanerstellung ist die zeitliche Dichte der angebotenen Kurse. Tabelle 2 zeigt die Verteilung der Lehrveranstaltungen über die Wochentage.

Wochentag	**Anteil (%)**
Montag	21.1%
Dienstag	25.4%
Mittwoche	21.8%
Donnerstag	19.7%
Freitag	11.3%
Samstag	0.7%
Sonntag	0%
Total	**100.0%**

Table 2: Prozentuale Verteilung der Kurse auf Wochentage

Wie aus den Daten ersichtlich wird, konzentriert sich der Großteil des Lehrangebots (ca. 88 %) auf die Kernzeit von Montag bis Donnerstag, während der Freitag mit 11.3 % bereits deutlich seltener belegt ist. Samstage spielen im regulären Lehrbetrieb nahezu keine Rolle. Diese ungleiche Verteilung verschärft die Problematik zeitlicher Überschneidungen ($\mathcal{F}_{Zeit}$) an Tagen mit besonders vielen Kursen.

Zusätzlich zur Verteilung auf die Tage ist die zeitliche Platzierung innerhalb eines Tages entscheidend für die Modellierung von Präferenzen. Tabelle 3 schlüsselt die Startzeiten der einzelnen Kurseinheiten auf.

Stunde	**Anteil (%)**
7:00	0.6%
8:00	16.9%
9:00	8.3%
10:00	10.1%
11:00	6.1%
12:00	8.0%
13:00	11.4%
14:00	8.1%
15:00	5.9%
16:00	10.3%
17:00	5.6%
18:00	8.8%
19:00	0.0%

Table 3: Prozentuale Verteilung der Start-Uhrzeit der Kurse

Die Analyse der Startzeiten zeigt, dass die WU Wien ein sehr breites Zeitfenster nutzt. Ein signifikanter Anteil der Kurse beginnt bereits um 08:00 Uhr (16.9 %), was für Studierende mit Präferenzen für einen späteren Vorlesungsbeginn (vgl. Szenario 4.4.1) eine hohe Anzahl an potenziell zu vermeidenden Einheiten bedeutet.

Neben dem Beginn der Einheiten variiert auch die Dauer der einzelnen Termine, was die Berechnung von Überschneidungsfreiheiten komplexer gestaltet als bei starren Zeitslots. Tabelle 4 stellt die Dauer der Einheiten dar. Nicht ganzstündige Einheiten wurden kaufmännisch gerundet.

Stunden	**Anteil (%)**
1	0.7%
2	27.1%
3	41.9%
4	27.2%
5	1.8%
6	0.8%
7	0.1%
8	0.3%

Table 4: Prozentuale Verteilung der Dauer der Kurse

Die überwiegende Mehrheit der Veranstaltungen (über 96 %) dauert zwischen zwei und vier Stunden. Da die Termine jedoch individuell als Zeitstempel (Start/Ende) vorliegen und nicht in ein fixes Raster gepresst sind, muss der Algorithmus die Überschneidung der Zeitintervalle berechnen, anstatt Slots zu vergleichen.

4 Analyse

4.1 Implementierung des ILP Baseline Ansatzes

Feldman und Golumbic vergleichen in Ihrem Paper [9] die Outputs der Algorithmen "Hill Climbing" und "Offering Order" mit dem optimalen Stundenplan, der mittels Brute-Force ermittelt wurde. Da ein Brute-Force Algorithmus bei der großen Anzahl der Vorlesungen im VVZ der WU eine zu lange Laufzeit hätte, wurde eine Integer Linear Programming (ILP) Lösung als Benchmark gewählt, die ebenfalls unter Berücksichtigung aller Nebenbedingungen den optimalen Stundenplan erzeugt [16].

4.1.1 Variablen, Parameter und Zielfunktion

Zur formalen Definition des Modells werden die folgenden Mengen, Parameter und Entscheidungsvariablen verwendet:

- **Mengen:**
  - $\mathcal{O}$: Menge aller verfügbaren Kursangebote (*Offerings*).
  - $\mathcal{D}$: Menge aller Kalendertage, an denen Kurse stattfinden.
  - $\mathcal{G}$: Menge der Kursgruppen (z. B. verschiedene Parallelveranstaltungen desselben Kurses).
  - $\mathcal{G}_g \subseteq \mathcal{O}$: Teilmenge der Angebote, die zur Gruppe $g \in \mathcal{G}$ gehören.
  - $\mathcal{D}_d \subseteq \mathcal{O}$: Teilmenge der Angebote, die Termine am Kalendertag $d \in \mathcal{D}$ haben.
  - $\mathcal{F}_{\text{Zeit}}$: Menge aller Paare (i, j) von Angeboten, die eine zeitliche Überschneidung aufweisen.
  - $\mathcal{M}_{\text{Muss}}, \mathcal{M}_{\text{Blockiert}} \subseteq \mathcal{O}$: Mengen von Angeboten, die zwingend gewählt bzw. ausgeschlossen werden müssen.
- **Parameter:**
  - Mark(i): Der berechnete Nutzen (Score) für das Kursangebot i.
  - $C_{\min}, C_{\max}$: Unter- und Obergrenze für die Gesamtzahl der zu wählenden Kurse.
  - $D_{\min}, D_{\max}$: Unter- und Obergrenze für die Anzahl der Kurse an einem belegten Tag.
- **Entscheidungsvariablen:**

– $y_i \in \{0,1\}$: Binäre Variable; 1 wenn Kursangebot i gewählt wird, sonst 0.

– $u_d \in \{0,1\}$: Binäre Hilfsvariable; 1 wenn am Tag d mindestens ein Kurs belegt wird, sonst 0.

Mathematische Formulierung des ILP-Modells

$$\text{maximiere:} \quad \sum_{i \in \mathcal{O}} \text{Mark}(i) \cdot y_i \tag{1}$$

unter den Nebenbedingungen:

$$\sum_{i \in \mathcal{O}} y_i \geq C_{\min} \tag{2}$$

$$\sum_{i \in \mathcal{O}} y_i \leq C_{\max} \tag{3}$$

$$\sum_{i \in \mathcal{G}_g} y_i \leq 1 \quad \forall g \in \mathcal{G} \tag{4}$$

$$\sum_{i \in \mathcal{D}_d} y_i \geq D_{\min} \cdot u_d \quad \forall d \in \mathcal{D} \tag{5}$$

$$\sum_{i \in \mathcal{D}_d} y_i \leq D_{\max} \cdot u_d \quad \forall d \in \mathcal{D} \tag{6}$$

$$y_i + y_j \leq 1 \quad \forall (i,j) \in \mathcal{F}_{\text{Zeit}} \tag{7}$$

$$y_i = 1 \quad \forall i \in \mathcal{M}_{\text{Muss}} \tag{8}$$

$$y_i = 0 \quad \forall i \in \mathcal{M}_{\text{Blockiert}} \tag{9}$$

$$y_i, u_d \in \{0,1\} \quad \forall i \in \mathcal{O}, \forall d \in \mathcal{D} \tag{10}$$

Figure 2: Mathematische Formulierung des ILP-Modells zur Stundenplanoptimierung

4.1.2 Nebenbedingungen (Constraints)

- **Anzahl der Lehrveranstaltungen** (Gleichungen 2 und 3): Begrenzen die Gesamtzahl der gewählten Lehrveranstaltungen, wobei $C_{\min}$ und $C_{\max}$ die definierten Mindest- bzw. Maximalanzahlen darstellen.

- **Gruppen-Beschränkung** (Gleichung 4): Stellt sicher, dass aus jeder vordefinierten Gruppe $g \in \mathcal{G}$ (verschiedene *Offerings* desselben Kurses) maximal eine Lehrveranstaltung ausgewählt wird.

- **Tägliche Kursanzahl** (Gleichungen 5 und 6): Regelt die Anzahl der Kurse pro Kalendertag $d \in \mathcal{D}$. Durch die binäre Hilfsvariable u_d wird sichergestellt, dass an einem Tag entweder gar kein Kurs ($u_d = 0$) oder eine Anzahl zwischen $D_{\min}$ und $D_{\max}$ ($u_d = 1$) belegt wird.

- **Zeitliche Überschneidungen** (Gleichung 7): Verhindert die gleichzeitige Auswahl von Lehrveranstaltungen, deren Termine sich zeitlich überschneiden. Die Menge $\mathcal{F}_{\text{Zeit}}$ enthält alle Paare (i, j), die einen Konflikt verursachen.

- **Feste Zuordnungen** (Gleichungen 8 und 9):

  1. $\mathcal{M}_{\text{Muss}}$ (8): Erzwingt die Auswahl von Lehrveranstaltungen, die als zwingend notwendig (Priorität = 100) definiert sind.
  2. $\mathcal{M}_{\text{Blockiert}}$ (9): Verbietet die Auswahl von Lehrveranstaltungen, die nicht gewählt werden dürfen (Priorität = -100).

Durch das Lösen dieses Modells wird ein optimaler Satz an Entscheidungsvariablen y_i (Stundenplan) bestimmt, der die Zielfunktion (*Mark*) maximiert, während alle Nebenbedingungen erfüllt werden. Das Ergebnis ist der optimal mögliche Stundenplan gemäß der definierten Constraints.

4.1.3 CPU vs. GPU

Um der technologischen Entwicklung seit der Veröffentlichung der ursprünglichen Algorithmen Rechnung zu tragen, wurde die ILP-Lösung sowohl für CPU- als auch für GPU-Architekturen (NVIDIA CUDA) implementiert [16][17]. Das ermöglicht eine Bewertung der Rechenzeit unter modernen Bedingungen, die über die Möglichkeiten der ursprünglichen Studie hinausgehen.

4.2 Implementierung der Hill-Climbing Algorithmen

Basierend auf der Methodik von Feldman und Golumbic [9] wurden die Optimierungsalgorithmen *Hill Climbing v1* und *Hill Climbing v3* implementiert. Das Ziel dieser Algorithmen ist es, inkrementell, durch lokal optimale Entscheidungen einen Stundenplan zu finden, der die Zielfunktion (*Mark*) maximiert.

4.2.1 Hill Climbing v1

Der *Hill Climbing v1*-Algorithmus wählt bei jedem Schritt das beste verfügbare *Offering* aus **allen verfügbaren Offerings** gemessen am *Mark* des Stundenplans inklusive des gewählten *Offering* [18].

4.2.2 Hill Climbing v3

Der *Hill Climbing v3*-Algorithmus unterscheidet sich vom *v1*-Ansatz dadurch, dass der Suchraum für einen neuen Kurs reduziert wird. Anstatt alle verfügbaren *Offerings* zu evaluieren, beschränkt *v3* die Auswahl auf eine Teilmenge der nach *Mark* sortierten Liste. Konkret wird in der Implementierung nur die zweite Hälfte — die Offerings mit der individuell besten *Mark* — der verfügbaren *Offerings* betrachtet [19].

4.2.3 Suchstrategie

Ausgehend vom aktuellen Stundenplan wird für jeden möglichen nächsten Zustand die *Mark* berechnet, indem der Stundenplan ergänzt um ein weiteres *Offering* durch die Evaluationsfunktion bewertet wird. Anschließend wird jenes Lehrangebot ausgewählt, das zu der höchsten Bewertung führt, und in den Stundenplan übernommen.

$$a^* = \arg \max_{a \in \text{available\_offerings}} (\text{get\_schedule\_mark}(\text{schedule} \cup \{a\})) \tag{11}$$

Nur jene *Offerings* werden in die Auswahl aufgenommen, die keine Nebenbedingungen verletzen (zeitliche Überschneidung, Planpunkt bereits erfüllt, *Offering* bereits gewählt, *Offering* in Liste verbotener Kurse). Die Schleife wird fortgesetzt, solange ein gültiges *Offering* gefunden wird, das zu einer Verbesserung oder Erweiterung des Stundenplans führt.

4.2.4 Abbruchkriterien

Der Algorithmus bricht ab, wenn einer der folgenden Zustände eintritt:

1. Der Stundenplan ist valide und hat die maximale Anzahl von Lehrveranstaltungen erreicht.

2. Der Stundenplan ist valide, und es kann keine weitere gültige *Offering* gefunden werden, ohne dass eine Nebenbedingung verletzt wird.

3. Es kann keine weitere *Offering* gefunden werden, die keine Constraints verletzt. Wenn die minimale Anzahl der Lehrveranstaltungen noch nicht erfüllt ist, führt das zu einem ungültigen Ergebnis.

Algorithm 2 Hill Climbing

```
1: schedule ← ∅
2: availableOfferings ← FilterConflicts(offerings)
3: while availableOfferings ≠ ∅ do
4:     nextOffering ← arg max_{o∈availableOfferings} Mark(schedule ∪ {o})
5:     if IsValid(schedule ∪ {nextOffering}) then
6:         schedule ← schedule ∪ {nextOffering}
7:     end if
8:     if |schedule| = MaxScheduleLength then
9:         return schedule
10:    end if
11:    availableOfferings ← FilterConflicts(offerings)
12: end while
13: return schedule
```

4.2.5 Zur Nicht-Implementierung von Hill Climbing v2

Der von Feldman und Golumbic vorgeschlagene *Hill Climbing v2*-Algorithmus priorisiert zusätzlich auf der Ebene von *Gruppen* (*Groups*). Im Originalkontext des Papers beziehen sich Gruppen auf die Unterscheidung zwischen Pflichtveranstaltungen (*Mandatory*) und Wahlveranstaltungen (*Elective*), wobei die Priorität der Gruppen die Auswahl von Pflichtveranstaltungen vor Wahlveranstaltungen erzwingt [9]. Da es im Hauptstudium Wirtschaftsinformatik keine Unterscheidung zwischen Pflicht- und Wahlveranstaltungen gibt, wäre die Implementierung funktional identisch mit *v1*.

4.3 Implementierung des Offering-Order Algorithmus

Basierend auf der Methodik von Feldman und Golumbic [9] wurde der Optimierungsalgorithmus *Offering Order* implementiert, der auf einer heuristischen Sortierung basiert.

4.3.1 Sortierung

Der Algorithmus nutzt die berechneten *Marks*, um eine Suchreihenfolge festzulegen:

1. Alle *Offerings* innerhalb des selben Planpunkts werden absteigend nach *Mark* sortiert. Innerhalb eines Planpunkts wird somit das Offering mit dem höchsten *Mark* zuerst in Betracht gezogen.

2. Die Planpunkte selbst werden nach dem *Mark* des besten *Offering* innerhalb des Planpunkts sortiert.

4.3.2 Suchstrategie

Der optimale Stundenplan wird durch eine *Forward-Checking Backtracking* Strategie gesucht. Es wird folgendermaßen vorgegangen:

1. Die Planpunkte werden iterativ durchlaufen. Gestartet wird mit dem Planpunkt, der das *Offering* mit der höchsten *Mark* beinhaltet (siehe 4.3.1).

2. Für jeden Planpunkt wird versucht, das *Offering* mit der höchsten *Mark* auszuwählen. Bevor die Auswahl getroffen wird, wird geprüft, ob das Hinzufügen des *Offerings* zu dem aktuellen Teilstundenplan zu einer zeitlichen Überschneidung führt oder andere Constraints verletzt (**Forward-Checking**)

3. Führt das Hinzufügen eines *Offerings* zu keinem gültigen Endergebnis in den nachfolgenden Planpunkten, wird die Auswahl rückgängig gemacht und das nächstbeste *Offering* der aktuellen Gruppe wird getestet (**Backtracking**)

Algorithm 3 Offering Order Algorithm

```
1: schedule ← ∅
2: i ← 0
3: loop
4:     select an offering o ∈ groups_i
5:     if there is no o then
6:         backtrack to preceding state
7:     else
8:         schedule ← schedule ∪ {o}
9:         for all groups_j (i < j ≤ n) do
10:            remove from groups_j all violating offers
11:        end for
12:        if any groups_j becomes empty then
13:            backtrack to preceding state
14:        else
15:            i ← i + 1
16:            if i = n then
17:                return schedule
18:            end if
19:        end if
20:    end if
21: end loop
```

4.4 Szenariendefinition

Um die beschriebenen Algorithmen unter realistischen Bedingungen zu testen, wurden folgende Szenarien definiert:

4.4.1 Freier Tag

Laut einer Umfrage an der Hochschule Trier gaben 81% der Studierenden an, gerne einen komplett freien Tag zu haben, um einem Nebenjob nachzugehen. 13% der Studierenden gaben an, jeden Tag erst um 9:45 Uhr beginnen zu wollen, und 6% möchten einen oder mehrere freie Vormittage haben. Dabei wurde am häufigsten der Freitag (31%) vor dem Montag (6%) gewünscht. 64% der Studierenden wäre es egal, welchen Tag sie freibekommen [13]. Aus diesen Ergebnissen wurden folgende Szenarien abgeleitet:

- Keine Kurse an Montagen
- Keine Kurse an Freitagen
- Jeder Tag der Woche bis 9:45 frei
- Keine Kurse an Donnerstagen und Freitagen

4.4.2 Bliebte Zeiten

Laut einer weiteren Umfrage [7] bevorzugen 74% der Studierenden Prüfungen und Kurse, die über das Semester verteilt stattfinden. Teilt man den Tag in die drei Zeitslots Vormittag, Nachmittag und Abend ein, so war der Nachmittag der beliebteste Zeitslot, gefolgt vom Vormittag und dem Abend. Aus diesen Ergebnissen wurden folgende Szenarien abgeleitet:

- Erhöhte Priorität (+90) für Nachmittags, negative Priorität (−90) für Abend
- Erhöhte Priorität (+70) für Vormittag, neutrale Priorität (0) für Nachmittag, negative Priorität (−80) für Abend

4.4.3 Ähnlicher Problemstellungen

Hinzu kommen klassische Constraints, die in University-Course-Timetabling-Problemen häufig vorkommen [6]. Diese Constraints betreffen meist die universitäre Planung des Curriculums und die Zuteilung von Kursen zu Räumen, es gibt jedoch auch Constraints, die die Erstellung individueller Stundenpläne der Studierenden betreffen:

- Studierende müssen von 12:00 bis 13:00 Zeit für ein Mittagessen haben
- Pro Tag werden entweder keine oder mindestens 3 Stunden Vorlesungen besucht
- Pro Tag dürfen nicht mehr als 8 Stunden Vorlesungen besucht werden

4.4.4 Vergleichbarkeit

Zusätzlich zu den oben genannten Constraints sollen die Algorithmen auch in Extremfällen hinsichtlich ihrer Performance getestet werden. Daher werden ergänzend folgende Szenarien definiert, die in verwandten Arbeiten nicht explizit untersucht wurden, jedoch zur Vergleichbarkeit zwischen den Algorithmen sinnvoll sind:

- Alle Kurse können frei von Constraints gewählt werden.
- 25% - 75% aller Kurse können nicht belegt werden.
- 1 - 7 Kurse sind vorgegeben
- Keine Kurse zwischem 6:00 morgens und 13:00.
- Keine Kurse Montags, Dienstags und Mittwochs

4.5 Versuchsaufbau

Das Paper von Feldman und Golumbic [9] wählte einen Brute-Force-Algorithmus als Baseline für Performance-Vergleiche. Da ein Brute-Force Algorithmus bei der großen Anzahl der Vorlesungen im VVZ der WU eine zu lange Laufzeit hätte, wurde eine Integer Linear Programming (ILP) Lösung als neue Baseline gewählt. Diese gewährleistet, ebenso wie der Brute-Force-Ansatz für kleine Instanzen, die Ermittlung des optimalen Stundenplans (mit der höchsten *Mark*). Die Ergebnisse der heuristischen Algorithmen (*Hill Climbing v1*, *Hill Climbing v3*, *Offering Order*) werden folglich relativ zur optimalen Lösung des ILP-Ansatzes bewertet. Zusätzlich zu den heuristischen Algorithmen wird auch eine GPU-optimierte Variante des ILP-Ansatzes getestet.

Als Maß für die Schwierigkeit (*Difficulty*) der Probleminstanz wird in dieser Arbeit nicht die Laufzeit des Brute-Force-Algorithmus verwendet, sondern die Anzahl der zu planenden Kurse (N).

Die Algorithmen wurden anhand der folgenden Leistungskennzahlen evaluiert:

1. **Mark**: Die Qualität der gefundenen Lösung (in Prozent relativ zum Baseline Ansatz).
2. **Laufzeit**: Die zur Lösungsfindung benötigte Zeit.
3. **Speicherbedarf**: Der benötigte Hauptspeicher.

4.5.1 Zielfunktion

Die *Mark* dient als Bewertungsfunktion für die Qualität eines erstellten Stundenplans und entspricht der im Paper von Feldman und Golumbic [9] definierten Zielfunktion. Sie quantifiziert, wie gut ein Stundenplan die Präferenzen und Constraints der Studierenden erfüllt. Die Berechnung erfolgt ausschließlich für *gültige* Stundenpläne, d. h. solche, die keine restriktiven Constraints ($|p| = P$) verletzen.

Jede Präferenz oder Restriktion ist im Modell durch eine *Priorität* p im Wertebereich $[-P, P]$ definiert. Positive Prioritäten kennzeichnen wünschenswerte Eigenschaften (z. B. bevorzugte Lehrveranstaltungen oder Zeitfenster), negative Prioritäten hingegen unerwünschte. Die Stärke des Präferenzwerts ist proportional zum Absolutwert der Priorität.

Die *Mark* eines gültigen Stundenplans S ergibt sich aus der Summe der positiven Beiträge durch gewählte Kurse und den Abzügen für Verletzungen sogenannter *fixed* und *non-fixed* Constraints:

$$Mark(S) = \underbrace{\sum_{c \in C_S} q_c}_{\text{Kursprioritäten}} - \underbrace{\sum_{h \in H^{(f)}_{\text{violated}}} |p^{(f)}_h|}_{\text{Strafen für fixed-Constraints}} - \underbrace{\sum_{t \in T^{(n)}_{\text{violated}}} v_t \cdot |p^{(n)}_t|}_{\text{Strafen für non-fixed-Constraints}} , \tag{12}$$

wobei gilt:

- C_S: Menge aller im Stundenplan S enthaltenen Kurse.
- q_c: Priorität des Kurses c.
- $H^{(f)}_{\text{violated}}$: Menge aller Stunden, in denen ein *fixed*-Constraint verletzt wurde.
- $p^{(f)}_h$: Priorität der Stunde h. Eine Verletzung liegt vor, wenn
  - $p^{(f)}_h < 0$ und die Stunde aktiv ist (der Studierende hat in einer ungewünschten Stunde Unterricht), oder
  - $p^{(f)}_h > 0$ und die Stunde inaktiv ist (der Studierende hat in einer gewünschten Stunde keinen Unterricht).
- $T^{(n)}_{\text{violated}}$: Menge der verletzten *non-fixed*-Constraints.
- $p^{(n)}_t$: Priorität des Constraints t.
- v_t: Anzahl der verletzten Stunden (bzw. des Zeitumfangs), der durch Constraint t betroffen ist.

Je höher der Wert von $M(S)$, desto besser erfüllt der Stundenplan die angegebenen Präferenzen. Ziel der Optimierungsalgorithmen ist daher die Maximierung von $M(S)$.

4.5.2 Durchführung

Die experimentelle Evaluation der Algorithmen erfolgte durch systematische Benchmarks, die Performance und Lösungsqualität der implementierten Ansätze unter verschiedenen Szenarien und Schwierigkeitsgraden testet.

Die Experimente wurden auf einer Virtual Private Server (VPS)-Instanz des Anbieters Hetzner durchgeführt, welche über eine x86 AMD CPU verfügte. Das gewährleistet eine konsistente und dedizierte Umgebung für die Messungen.

Die Algorithmen wurden in TODO (N) verschiedenen Szenarien getestet. Jedes Szenario stellt eine vorkonfigurierte Zusammensetzung von Constraints dar (definiert in JSON-Konfigurationsdateien, z.B. `constraint1.json`).

Der Schwierigkeitsgrad (*Difficulty*) einer Probleminstanz wird durch die Anzahl der zu planenden Kurse N definiert, im Gegensatz zur Laufzeit des Brute-Force-Algorithmus, die in der Originalarbeit verwendet wurde. Die Tests wurden inkrementell für jede mögliche Kursanzahl N, beginnend bei $N = 1$ bis zur maximal möglichen Kursanzahl im jeweiligen Szenario, durchgeführt. Die maximale Kursanzahl wurde dabei vorab mithilfe des ILP-Baseline-Ansatzes ermittelt.

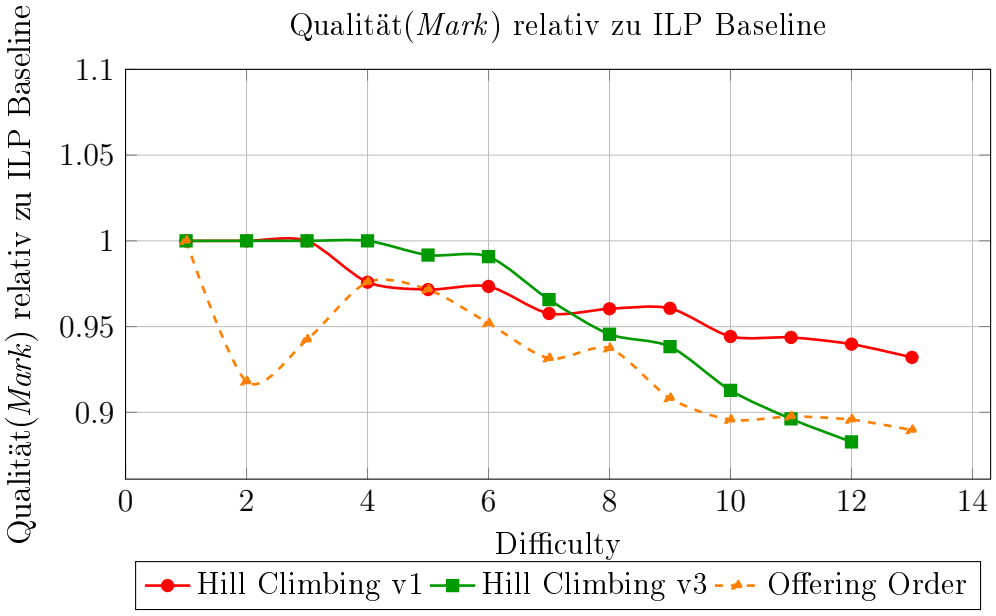
Für jeden Algorithmus ($A \in \{\text{ILP, Offering Order, Hill Climbing V1, Hill Climbing V3}\}$), für jede Kursanzahl N und für jedes Szenario S wurde die Messreihe wie folgt durchgeführt:

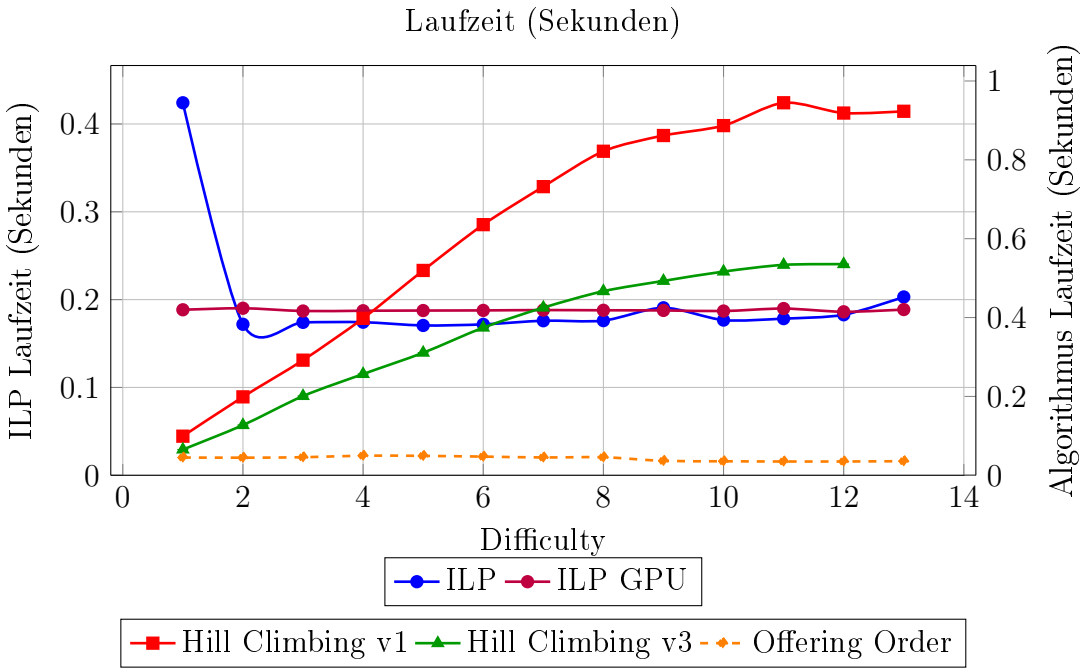
1. Jeder Algorithmus wurde für eine feste Kursanzahl N 50 Mal unabhängig voneinander ausgeführt.
2. Die 50 Messungen pro Algorithmus und Schwierigkeitsgrad wurden sequenziell durchgeführt, um sicherzustellen, dass die Messergebnisse für die Laufzeit und den Speicherbedarf unabhängig sind.
3. Es wurden folgende Metriken für jede der 50 Ausführungen erfasst:
   - Laufzeit (t): Die zur Lösungsfindung benötigte Zeit (in Sekunden).
   - Hauptspeicher (M): Der während der Ausführung belegte Hauptspeicher (in Megabyte).
   - Qualität (*Mark*)
4. Für die 50 Messwerte der Laufzeit und des Hauptspeichers wurden folgende statistische Kennzahlen berechnet:
   - Median ($t_{\text{median}}, M_{\text{median}}$)
   - Mittelwert ($t_{\text{mean}}, M_{\text{mean}}$)
   - Minimalwert ($t_{\text{min}}, M_{\text{min}}$)
   - Maximalwert ($t_{\text{max}}, M_{\text{max}}$)
   - Standardabweichung ($t_{\text{stdev}}, M_{\text{stdev}}$)

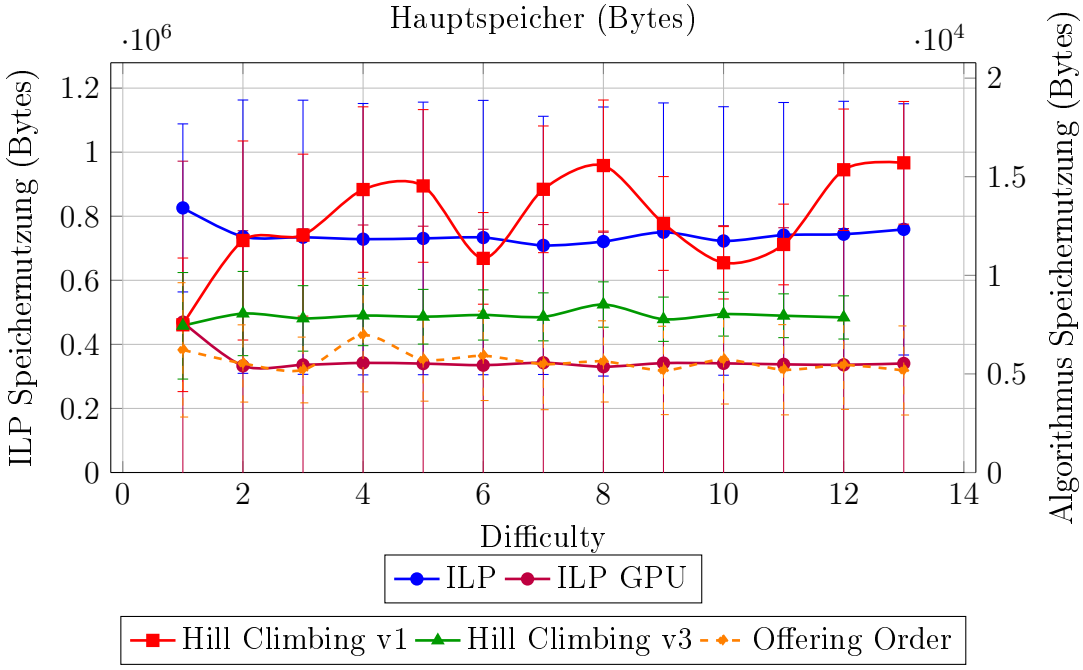
Um die Algorithmen vergleichbar zu halten, wurde der gemessene Zeit- und Speicherbedarf auf die Ausführung des Algorithmus selbst begrenzt. Das Preprocessing — insbesondere die Vorabberechnung der *Mark* für jedes Offering, die einmalig vor der Hauptschleife durchgeführt wird — ist nicht in den Messwerten enthalten.

5 Ergebnisse

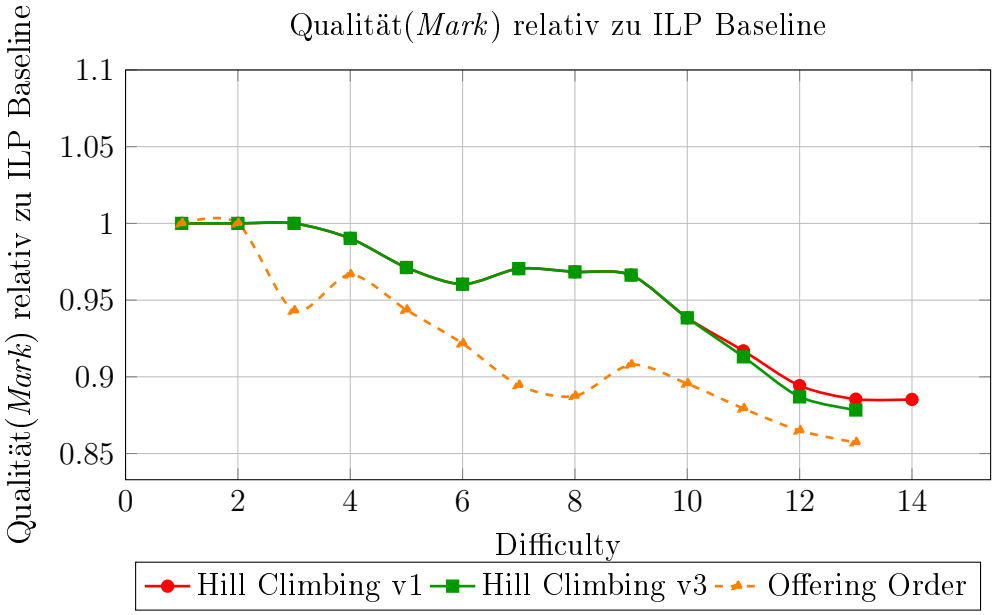
5.1 Szenario 1: Keine Kurse an Montagen

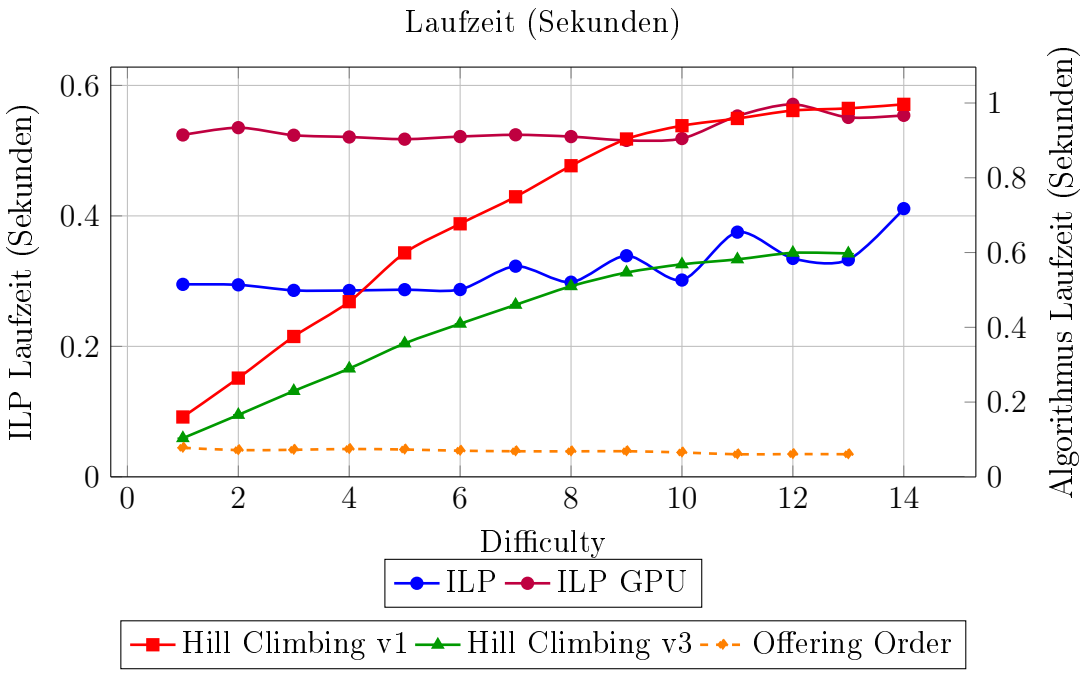


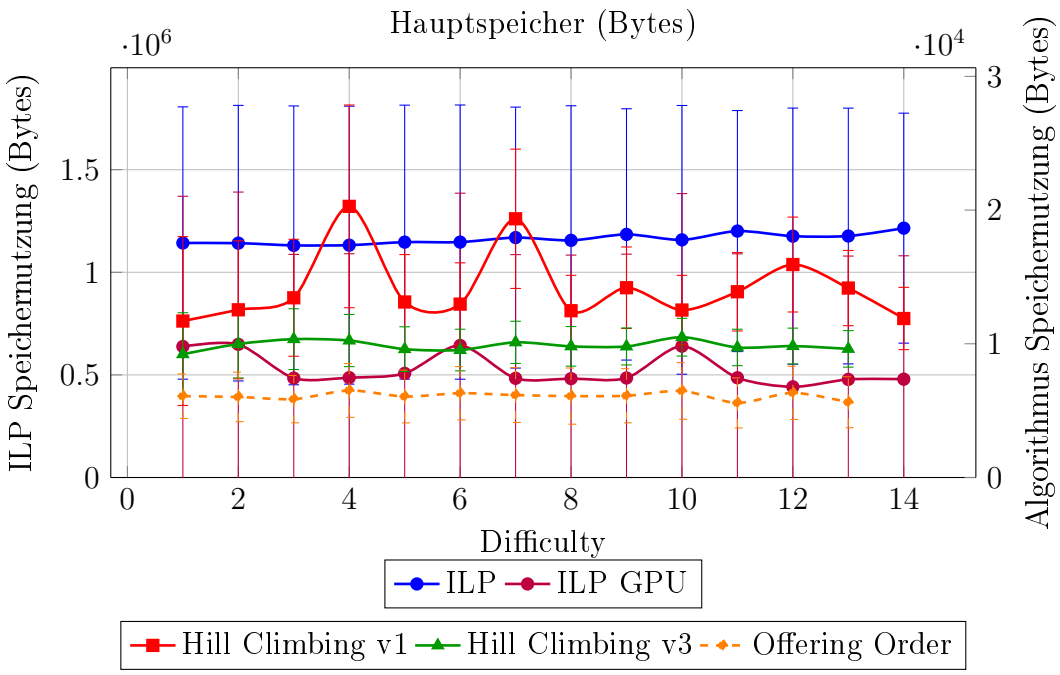



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
1.2
1
0.8
0.6
0.4
0.2
0
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

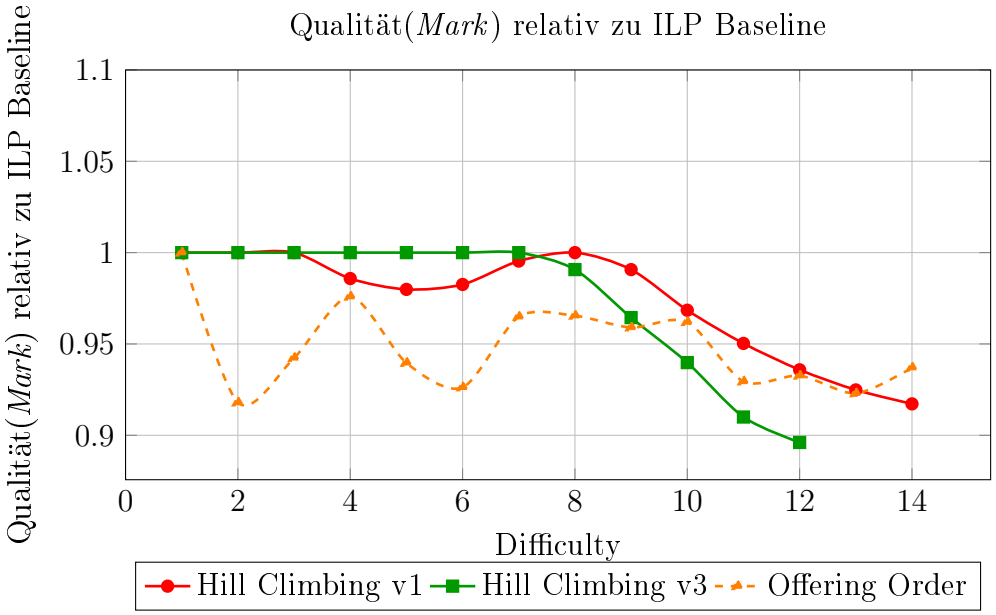
5.2 Szenario 2: Keine Kurse an Freitagen

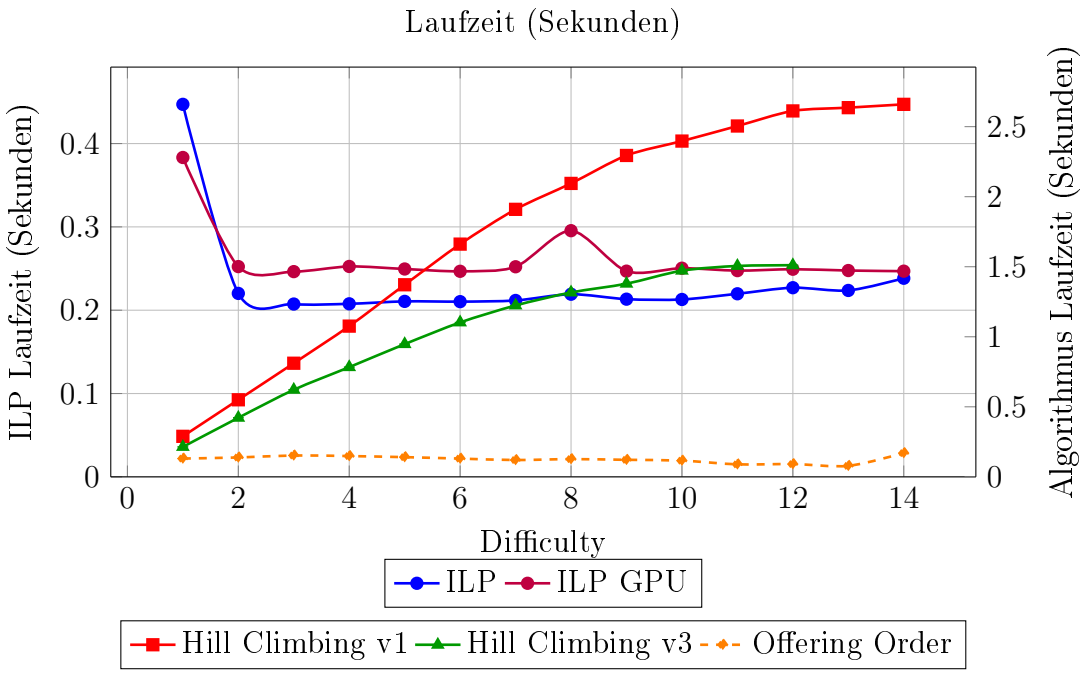


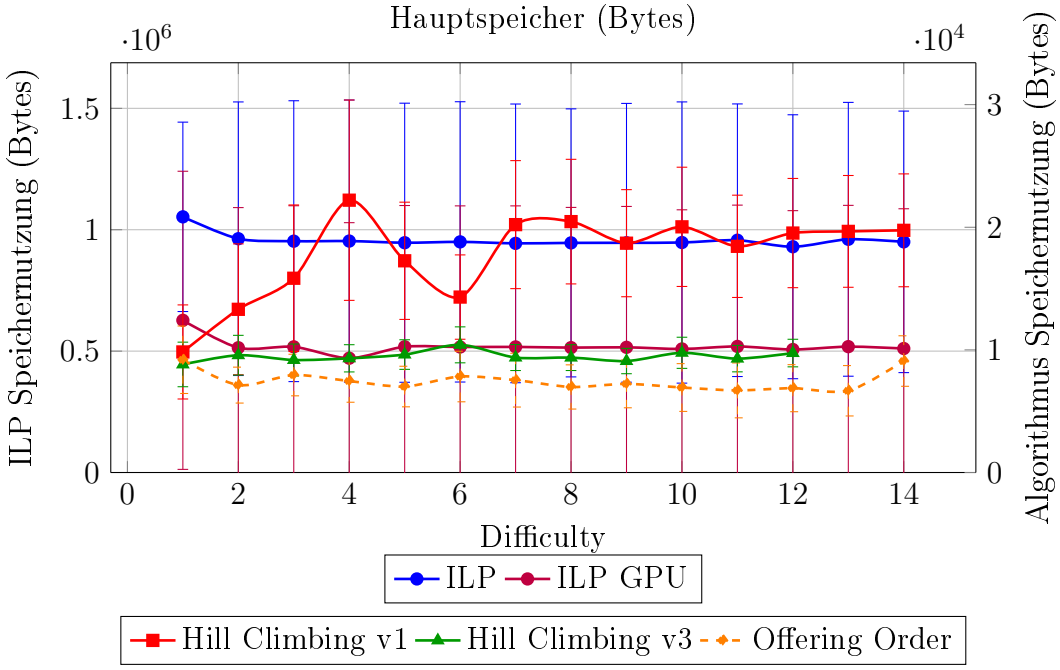



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
0.5
1
1.5
0
1
2
3
0
2
4
6
8
10
12
14
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

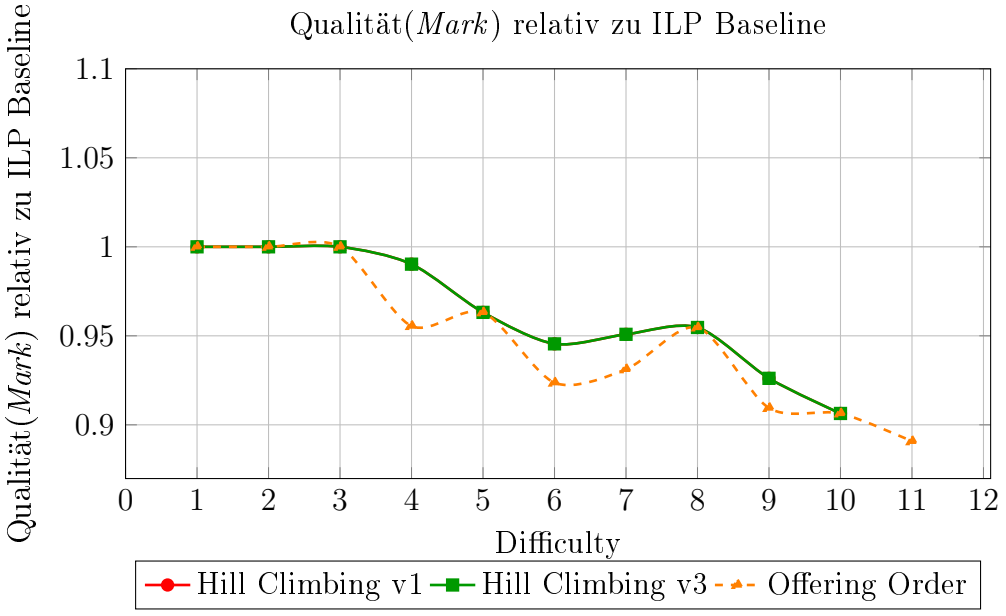
5.3 Szenario 3: Jeder Tag der Woche bis 9:45 frei

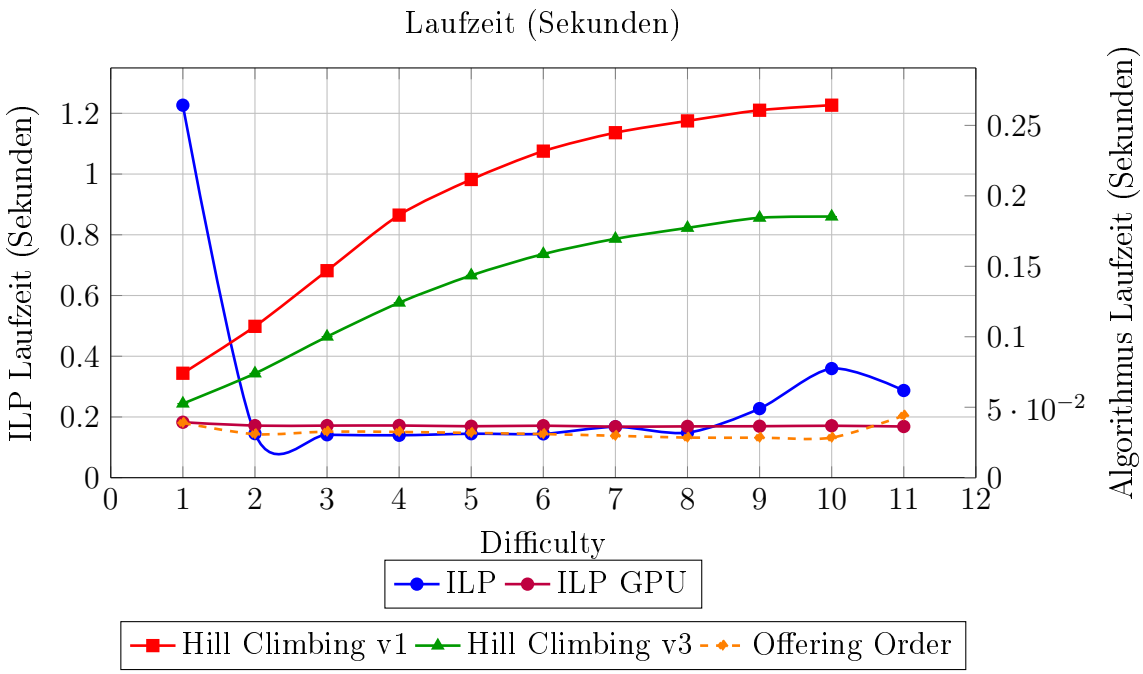


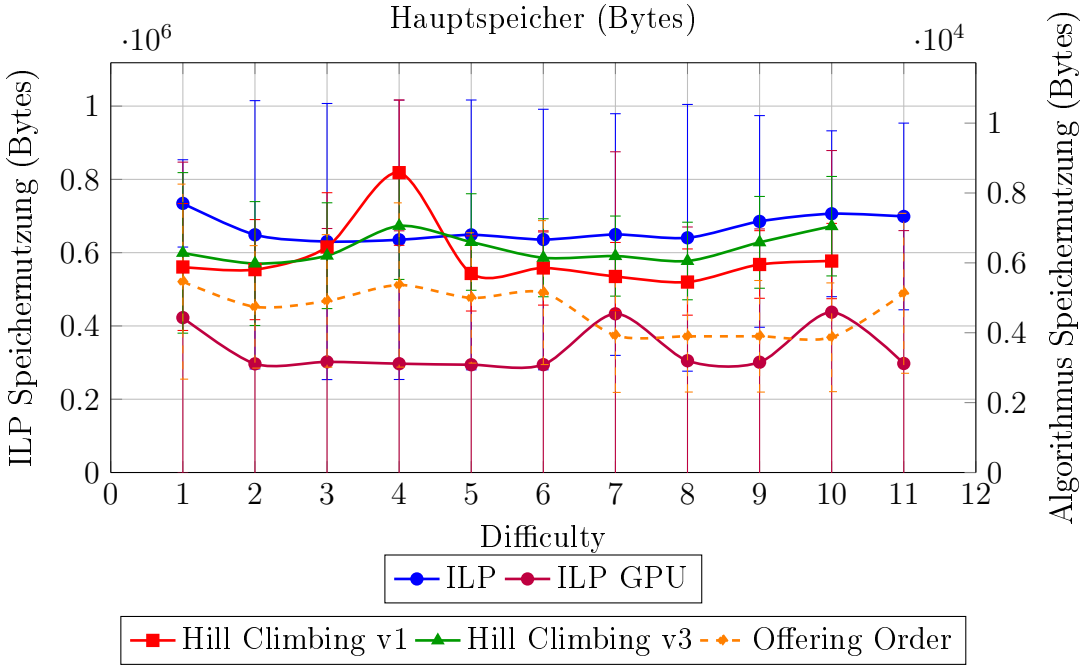



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
0.5
1
1.5
0
1
2
3
0
2
4
6
8
10
12
14
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

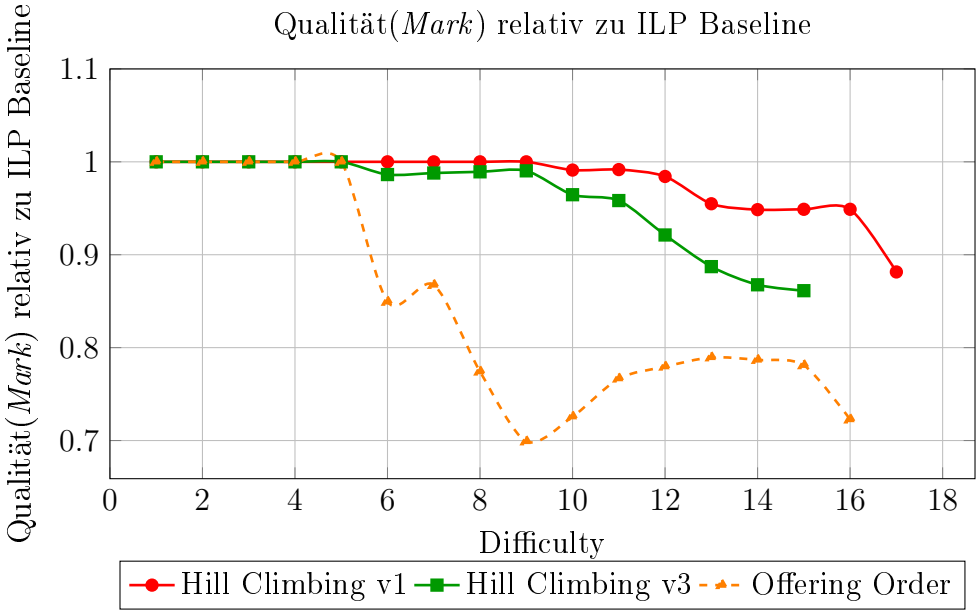
5.4 Szenario 4: Keine Kurse an Donnerstagen und Freitagen

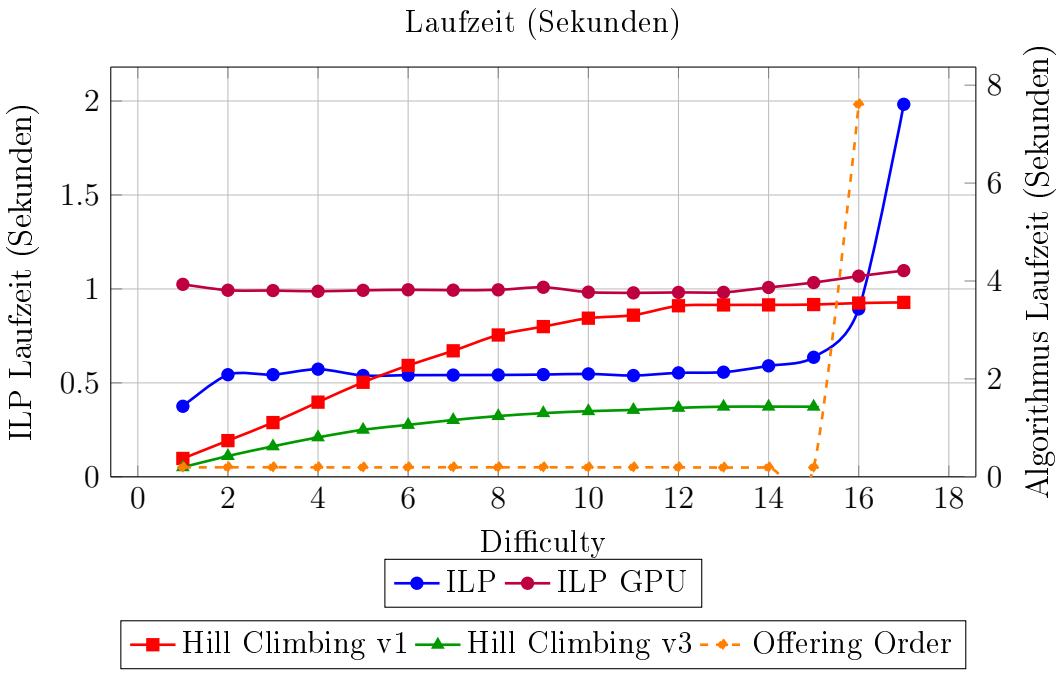


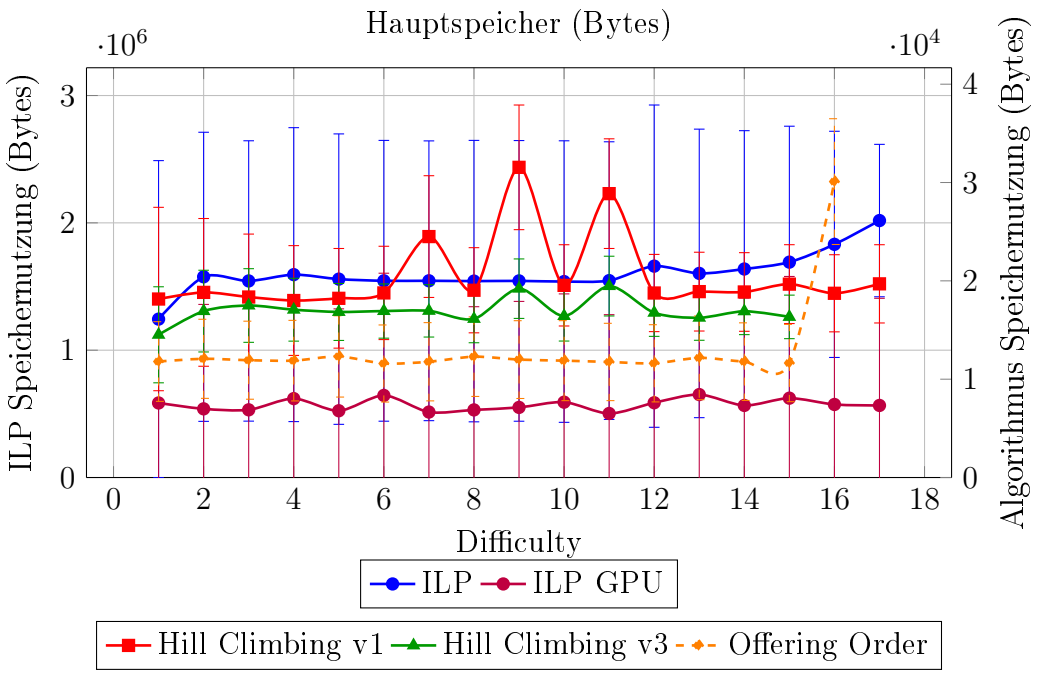



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5
6
7
8
9
10
11
12
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

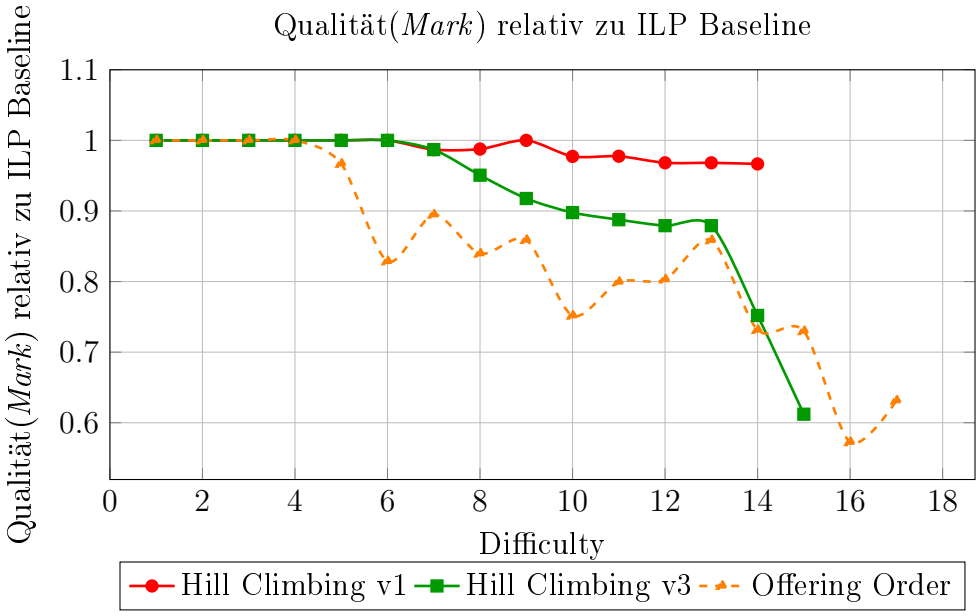
5.5 Szenario 5: Erhöhte Priorität (+90) für Nachmittags, negative Priorität (-90) für Abend

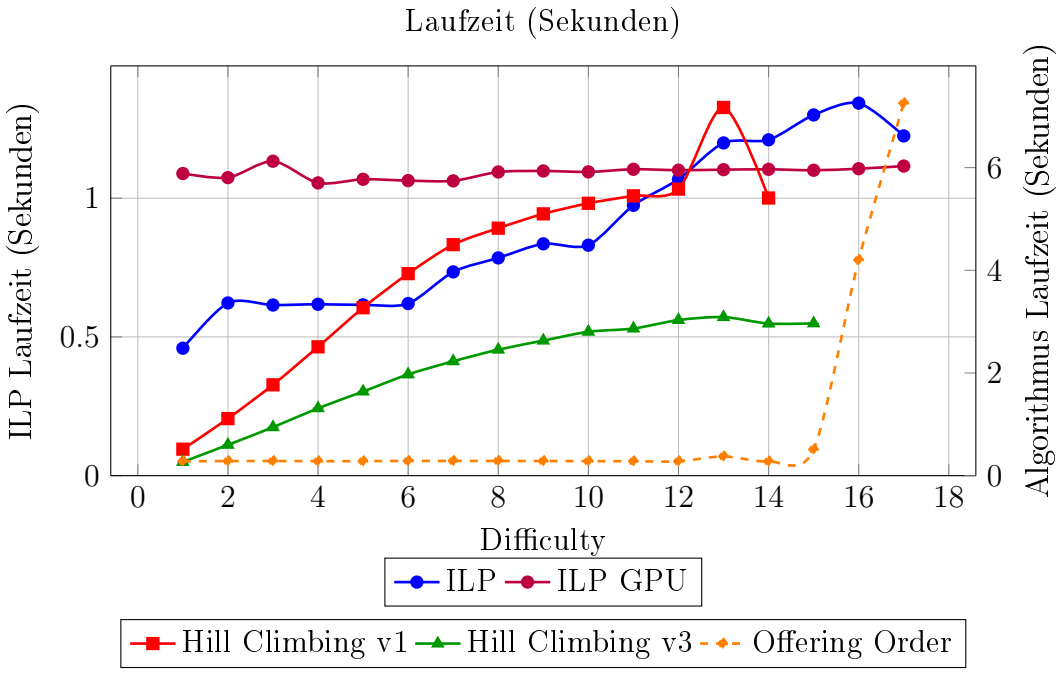


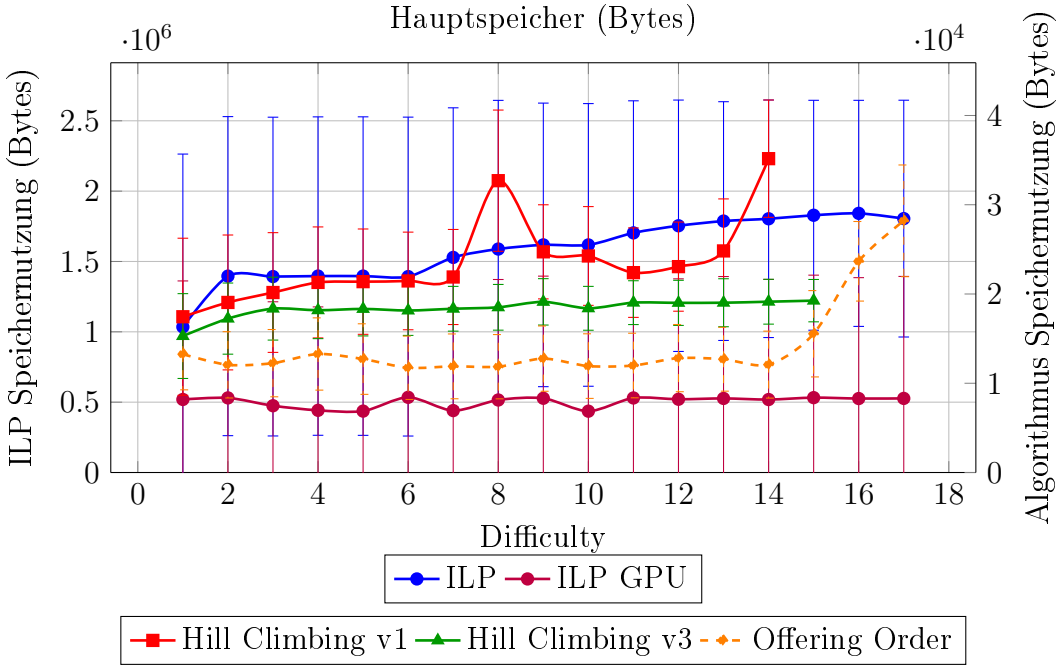



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
1
2
3
4
0
2
4
6
8
10
12
14
16
18
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

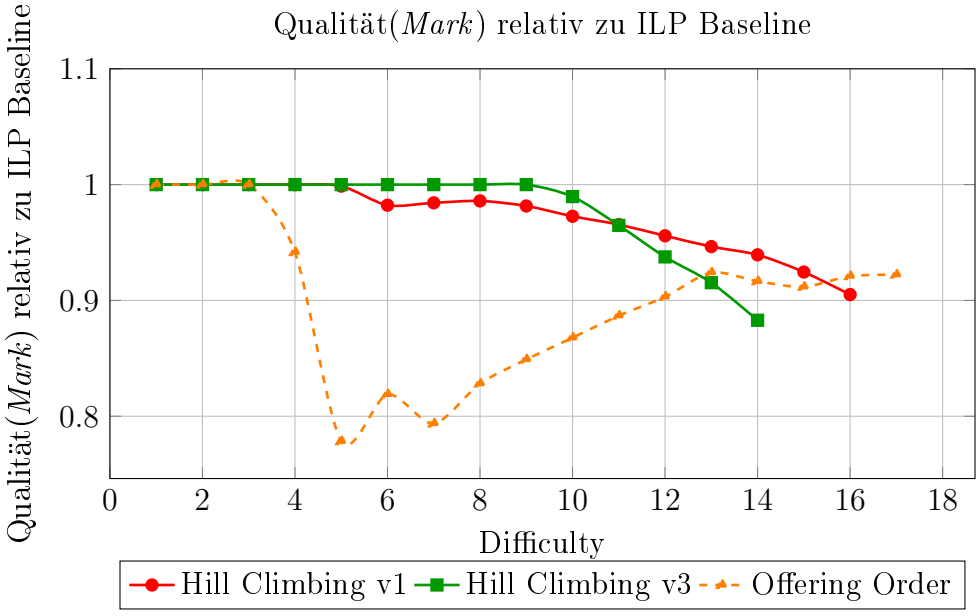
5.6 Szenario 6: Erhöhte Priorität (+70) für Vormittag, neutrale Priorität (0) für Nachmittag, negative Priorität (-80) für Abend

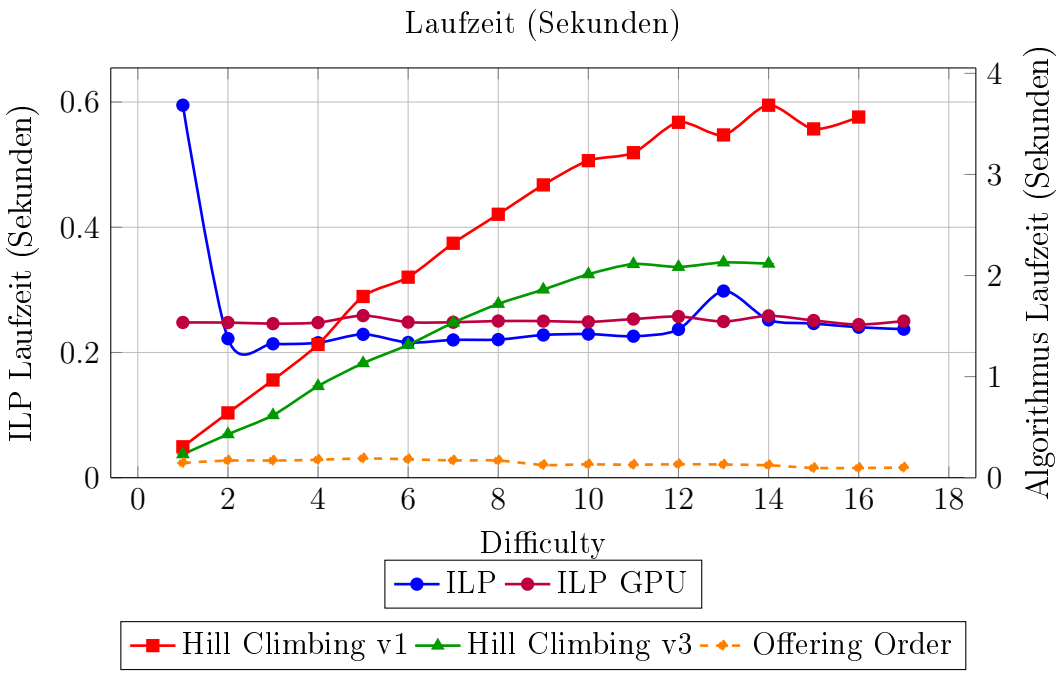


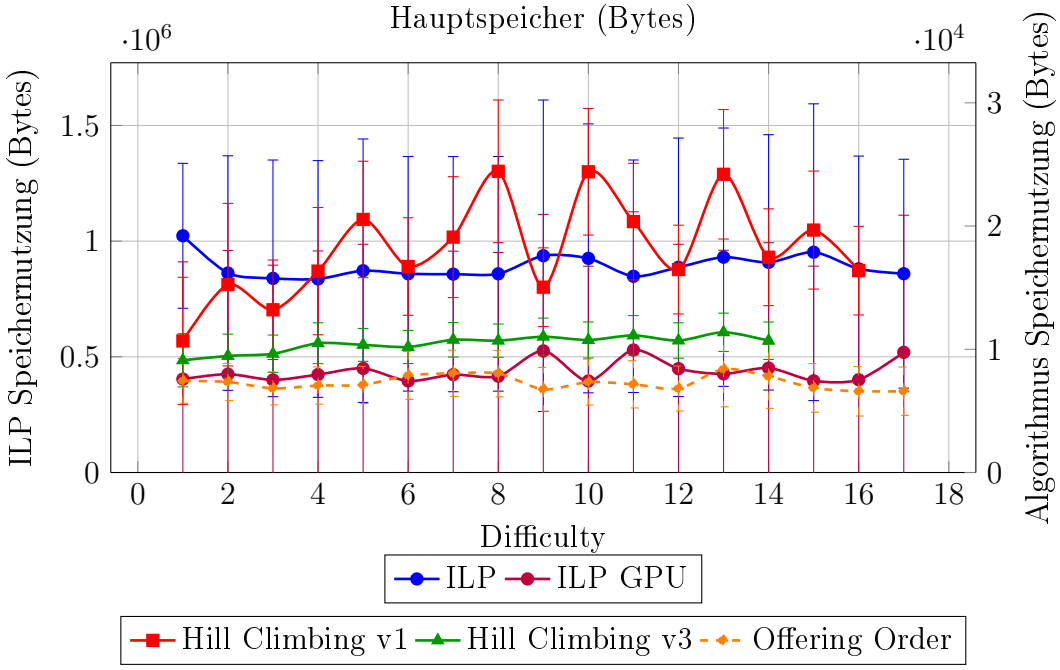



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
0.5
1
1.5
2
2.5
0
1
2
3
4
0
2
4
6
8
10
12
14
16
18
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

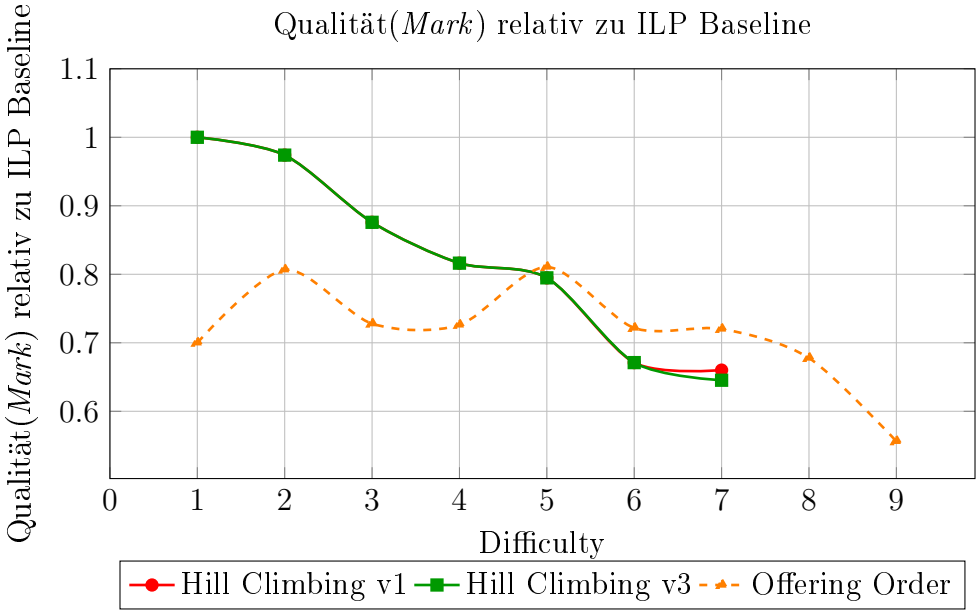
5.7 Szenario 7: Studierende müssen von 12:00 bis 13:00 Zeit für ein Mittagessen haben

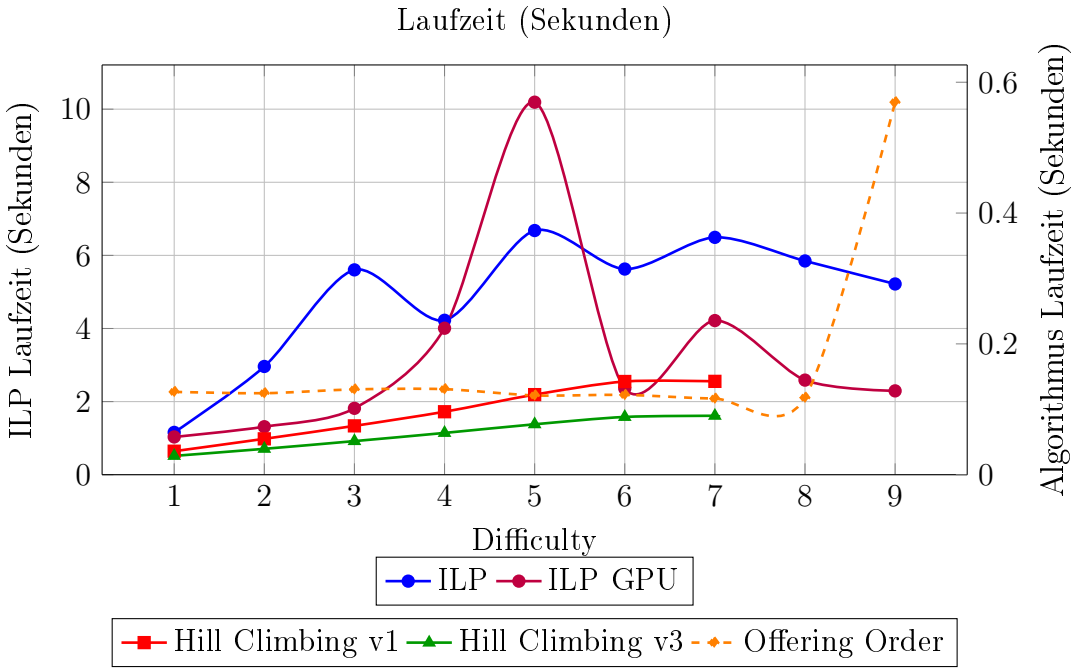


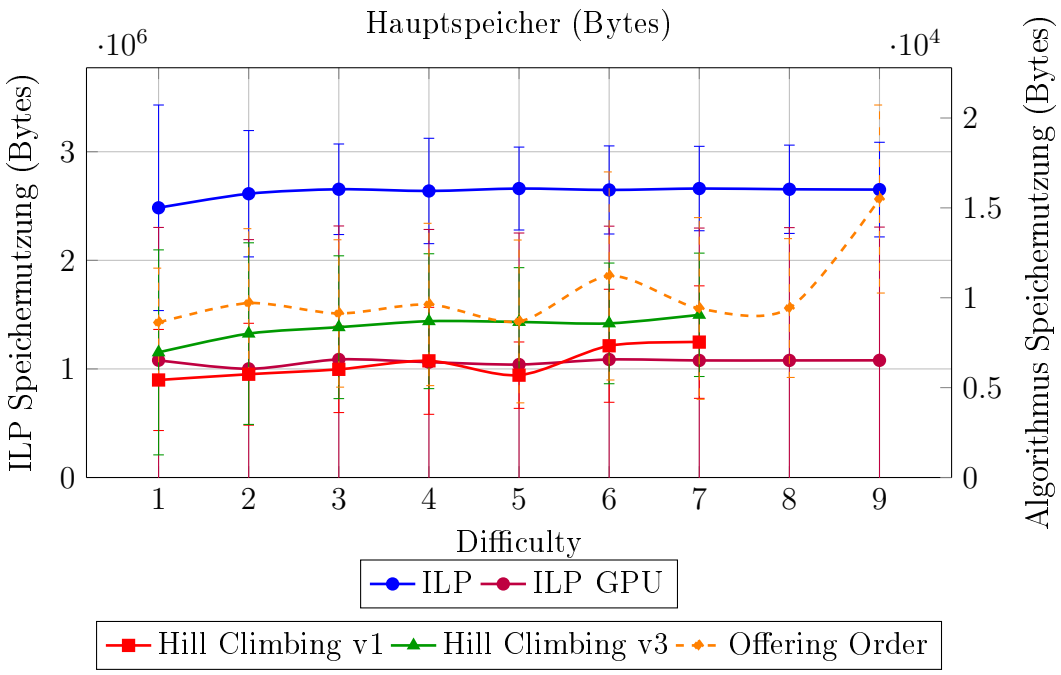



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
0.5
1
1.5
0
1
2
3
0
2
4
6
8
10
12
14
16
18
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

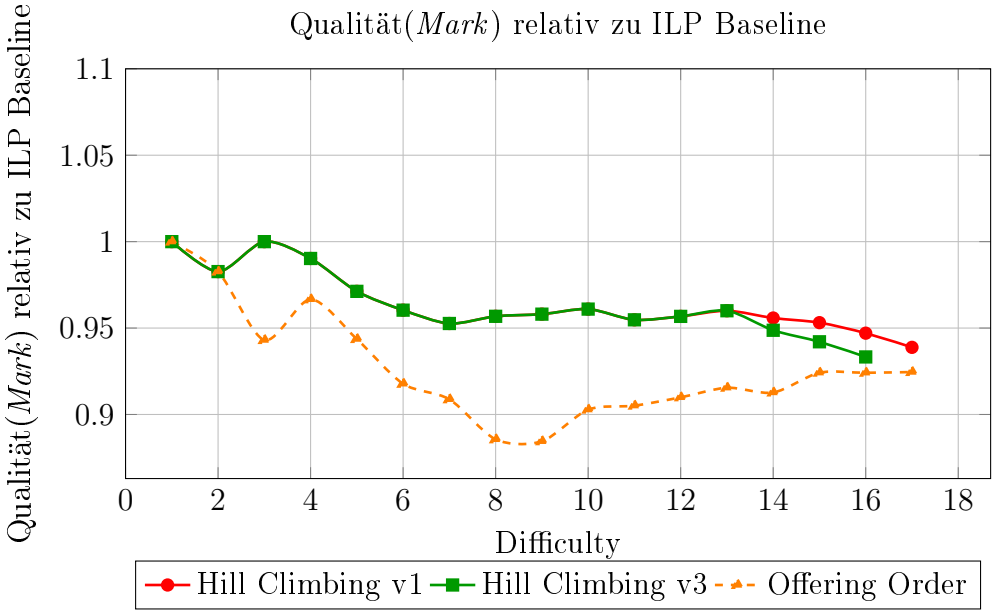
5.8 Szenario 8: Studierende müssen mehr als einen Kurs pro Tag besuchen

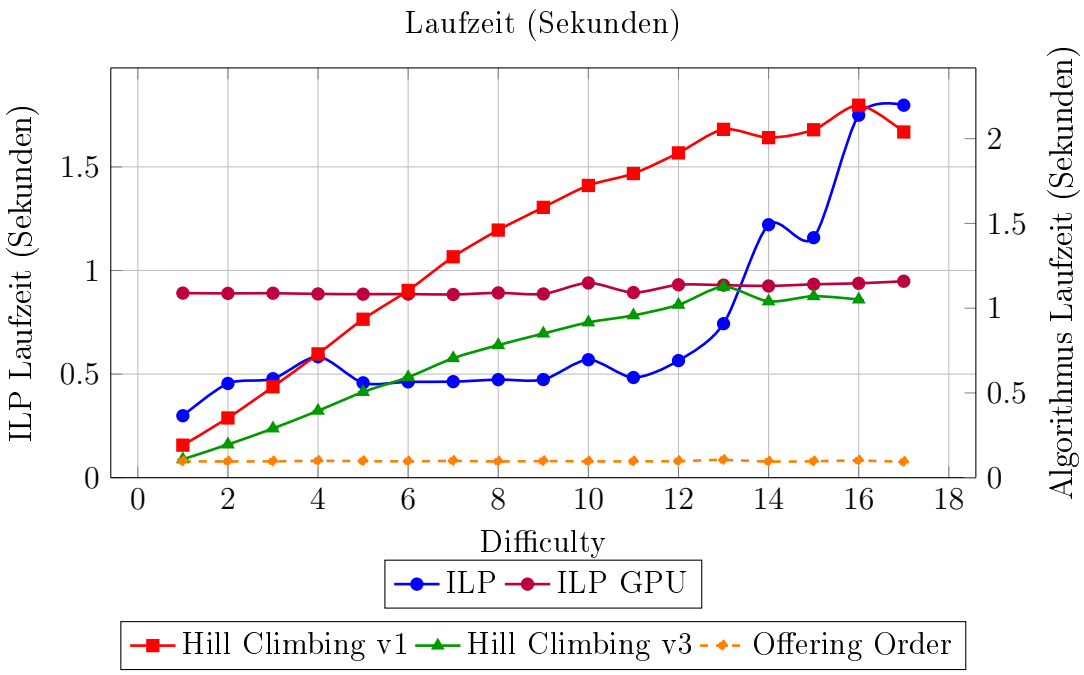


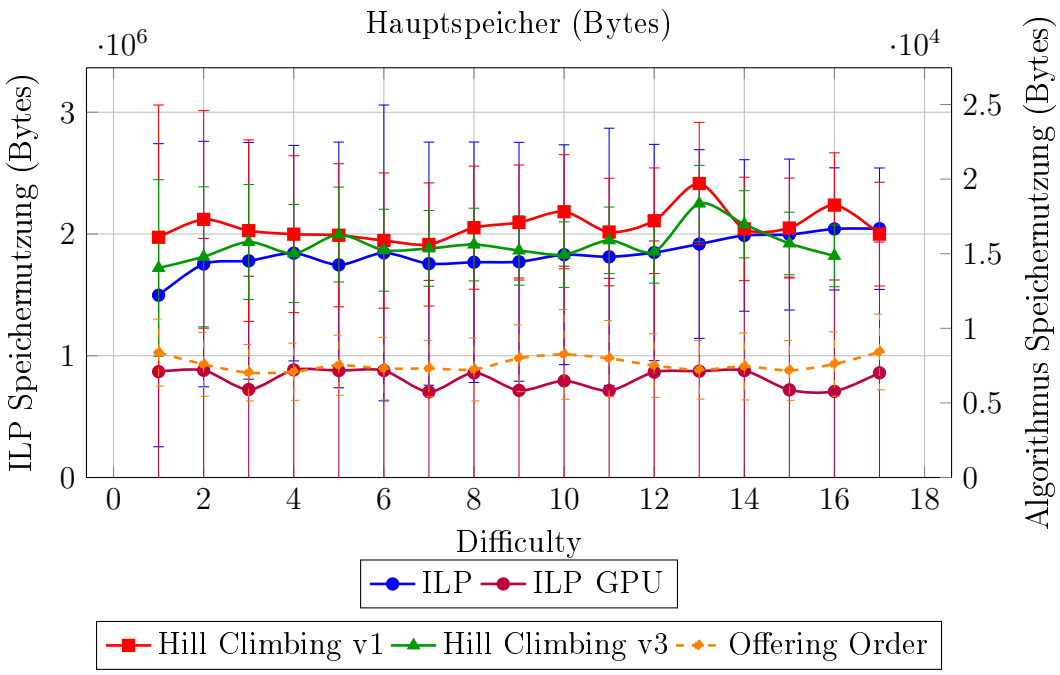



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
1
2
3
0
0.5
1
1.5
2
1
2
3
4
5
6
7
8
9
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

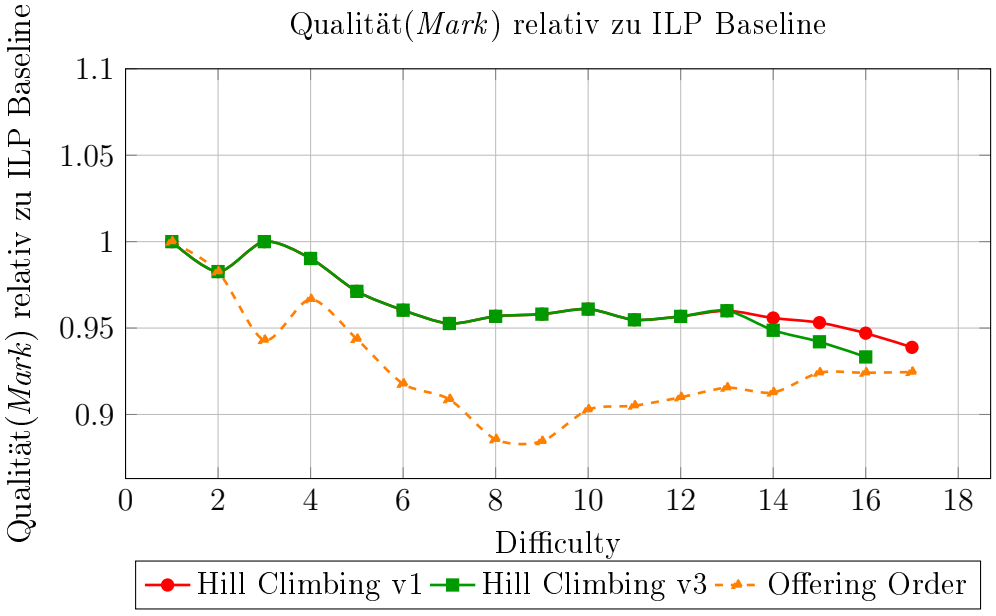
5.9 Szenario 9: Studierende dürfen nicht mehr als drei Kurse pro Tag besuchen

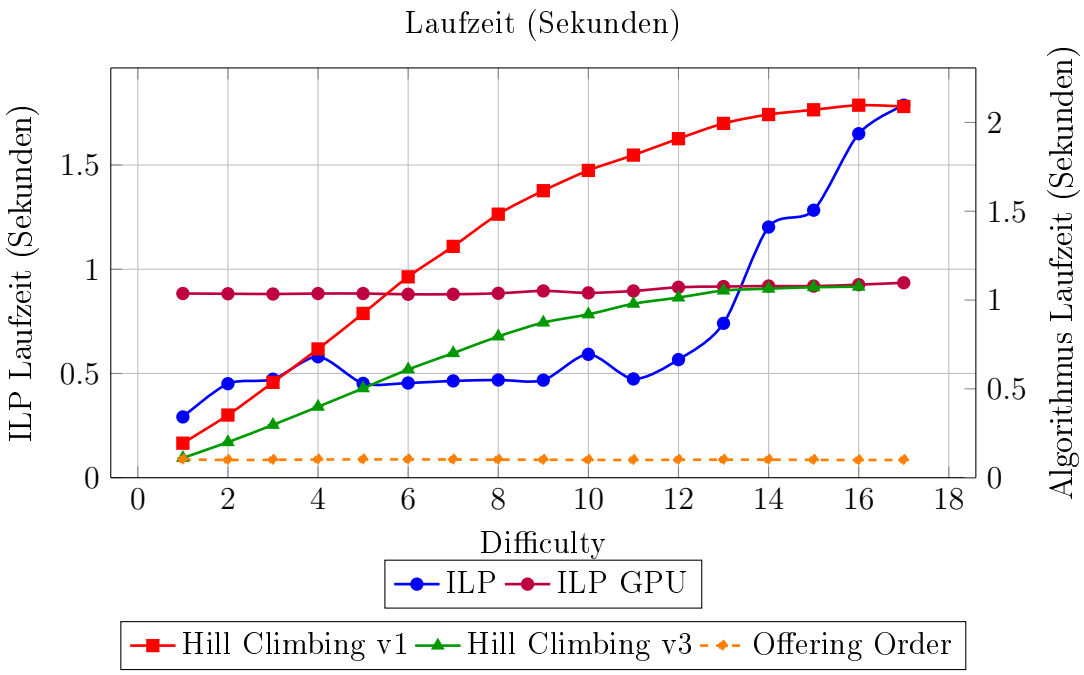


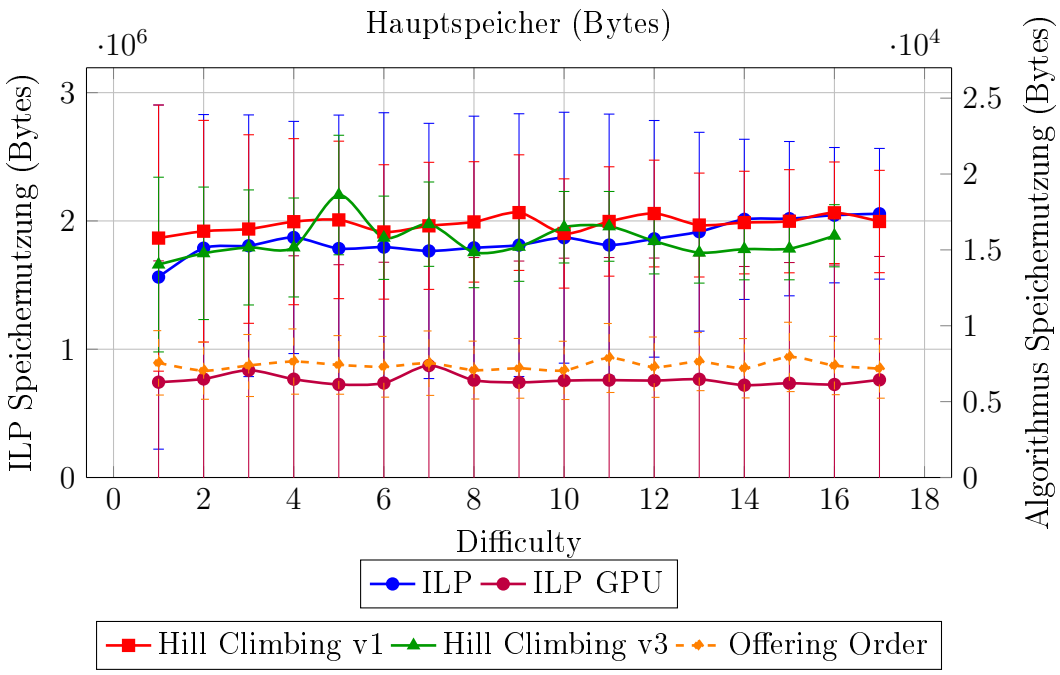



Hauptspeicher (Bytes)
·10^6
·10^4
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
1
2
3
0
0.5
1
1.5
2
2.5
0
2
4
6
8
10
12
14
16
18
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

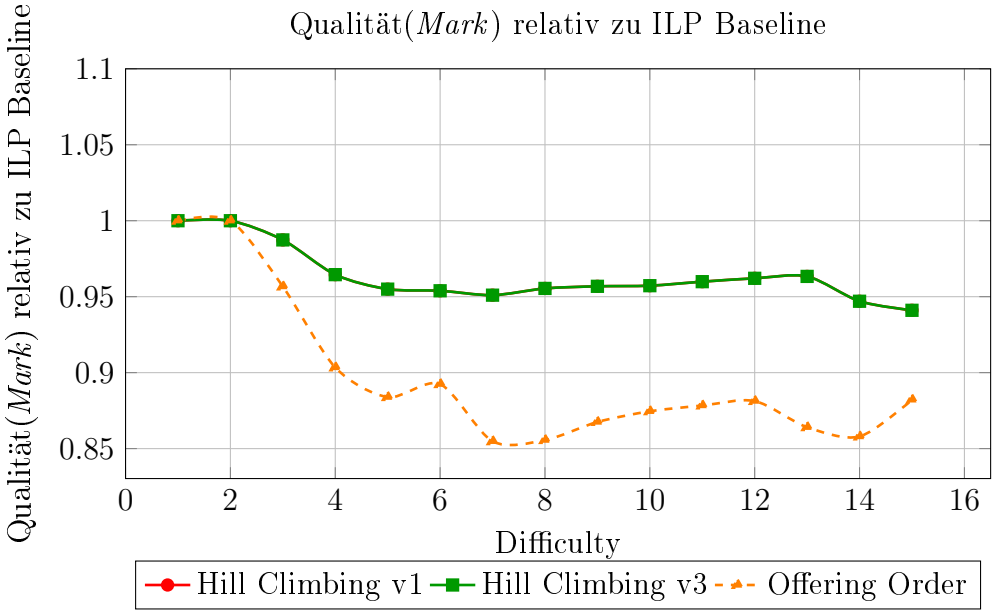
5.10 Szenario 10: Alle Kurse können frei von Constraints gewählt werden

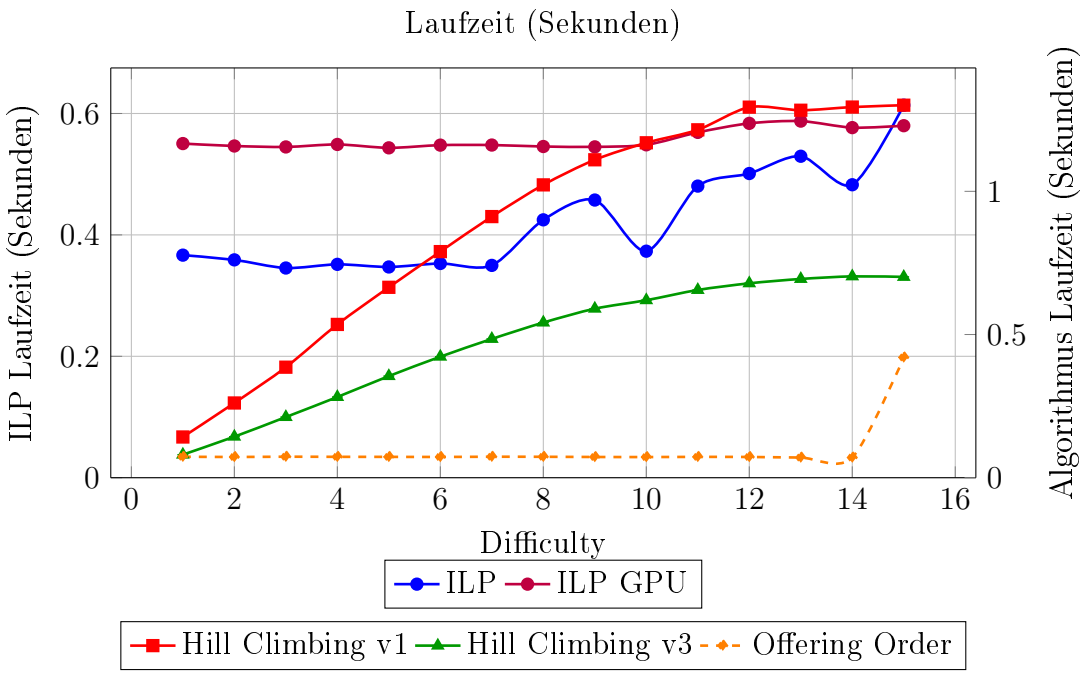


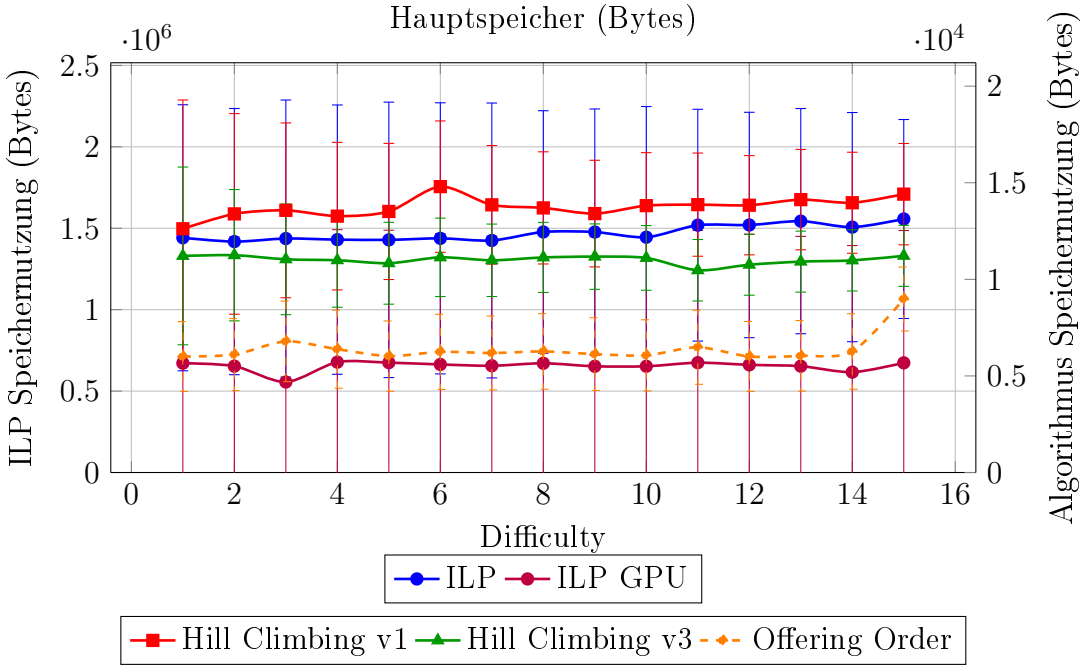



Hauptspeicher (Bytes)
$\cdot 10^6$
$\cdot 10^4$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
1
2
3
0
0.5
1
1.5
2
2.5
0
2
4
6
8
10
12
14
16
18
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

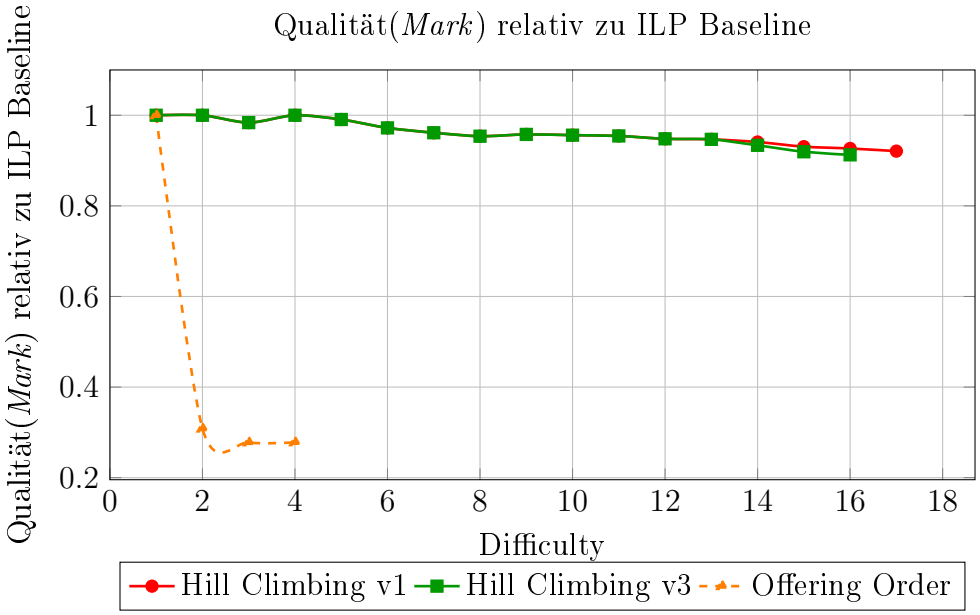
5.11 Szenario 11: 25%- 75% aller Kurse können nicht belegt werden.

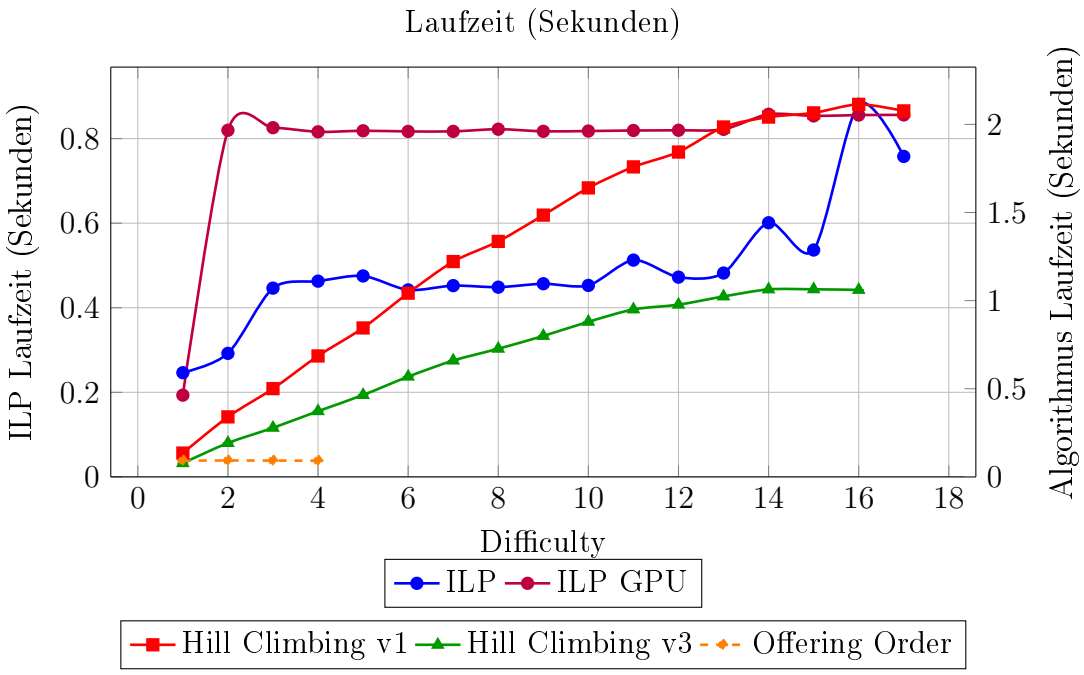


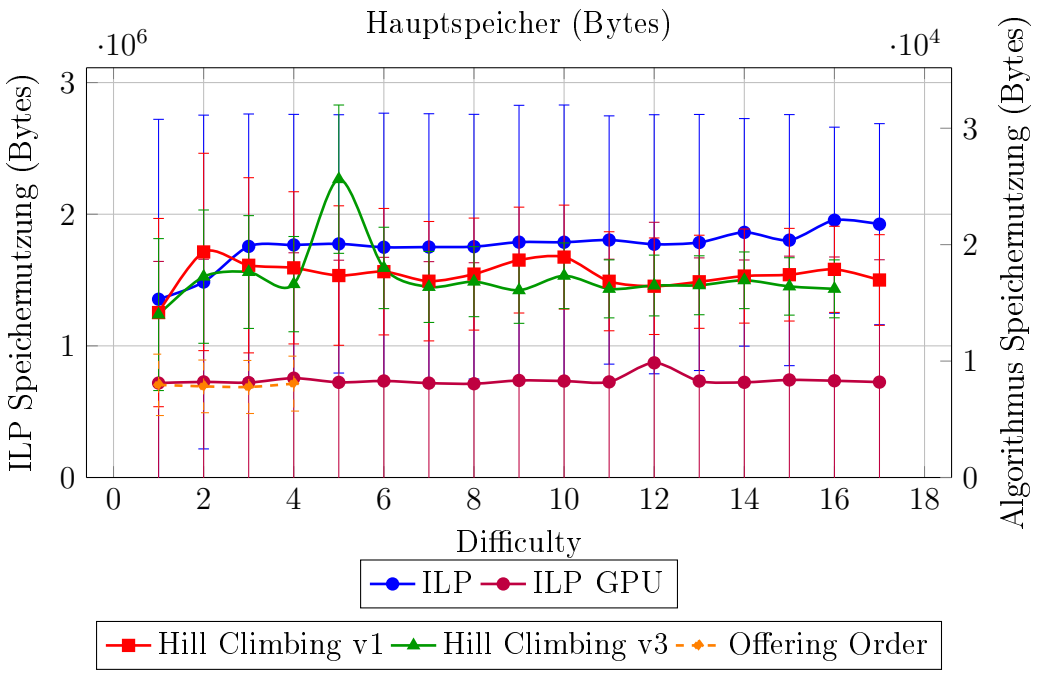



Hauptspeicher (Bytes)
·10^6
·10^4
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
16
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

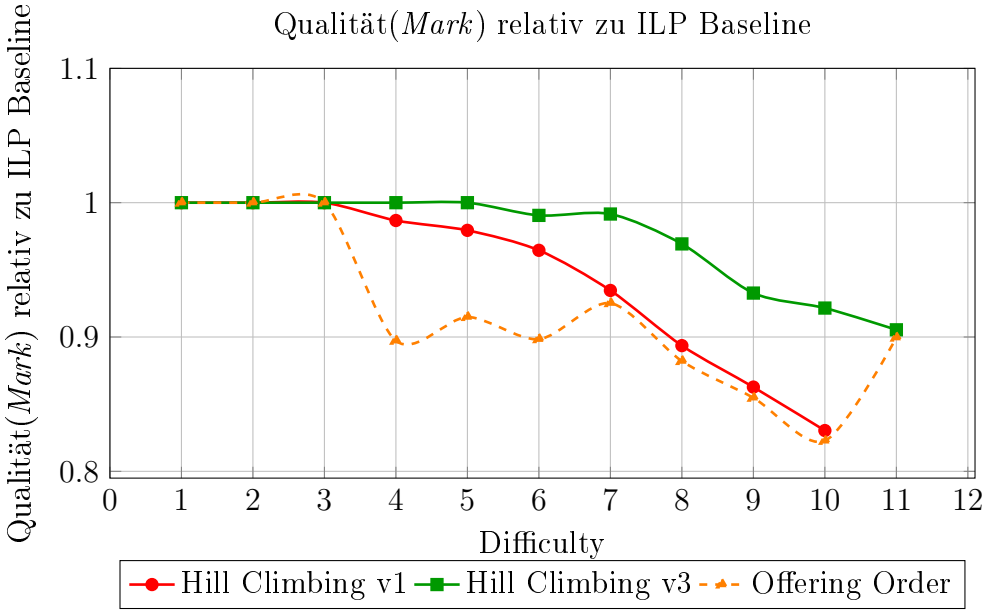
5.12 Szenario 12: 1 - 7 Kurse sind vorgegeben

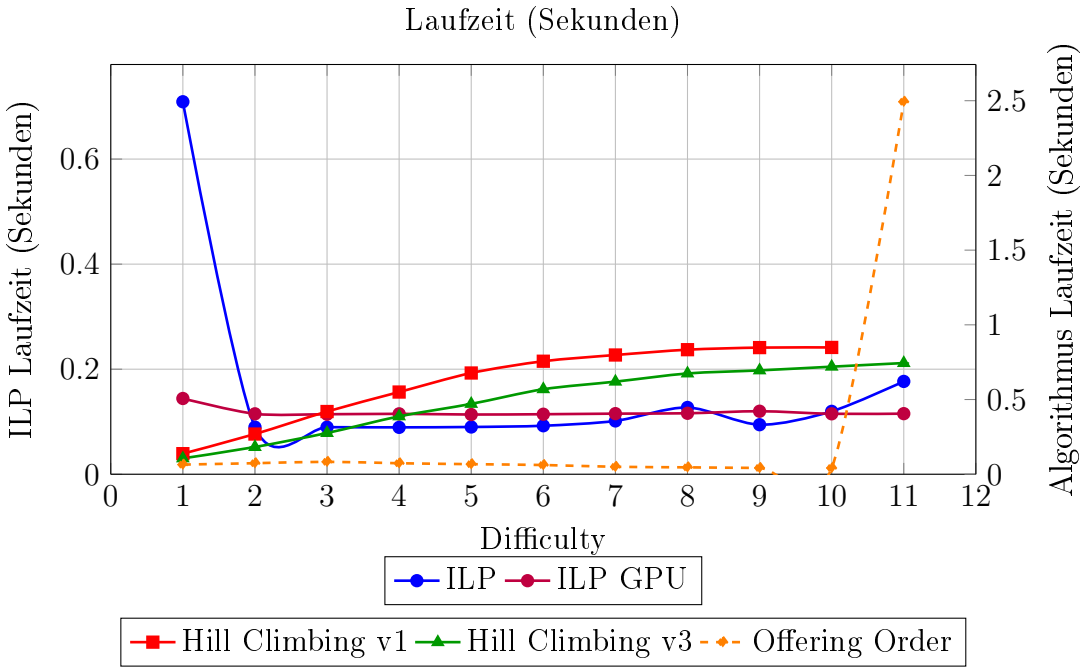


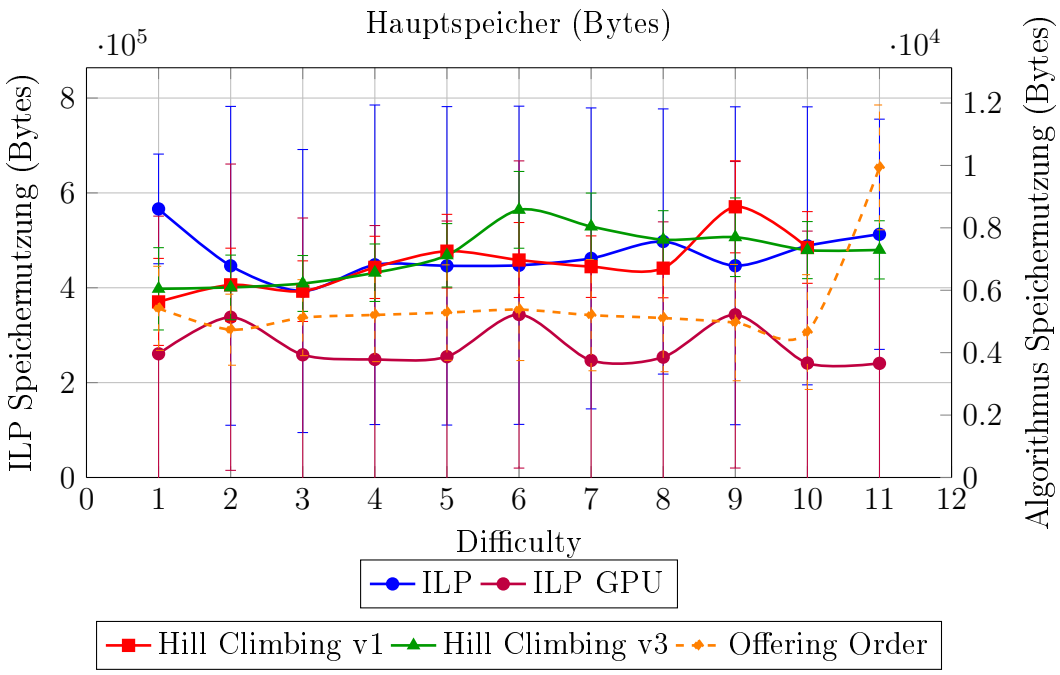



Hauptspeicher (Bytes)
·10^6
·10^4
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
1
2
3
0
2
4
6
8
10
12
14
16
18
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

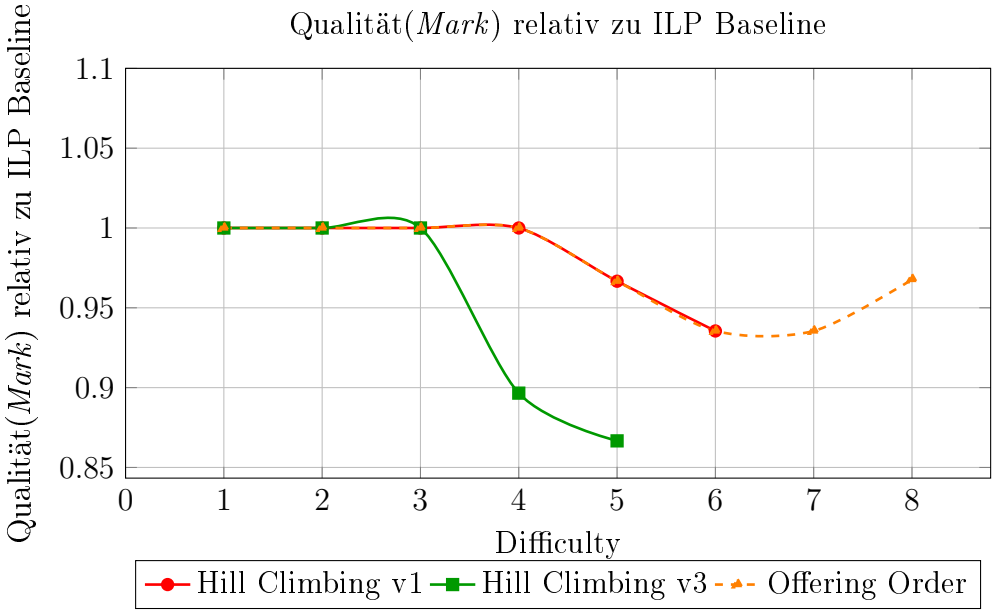
5.13 Szenario 13: Keine Kurse zwischem 6:00 morgens und 13:00

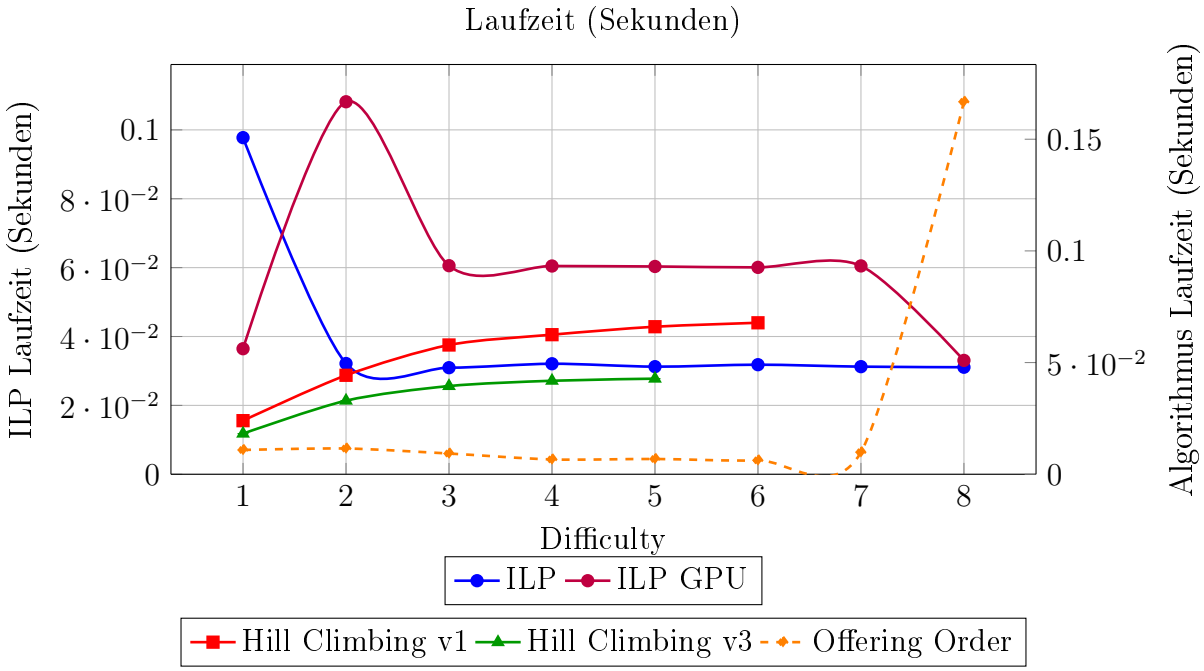


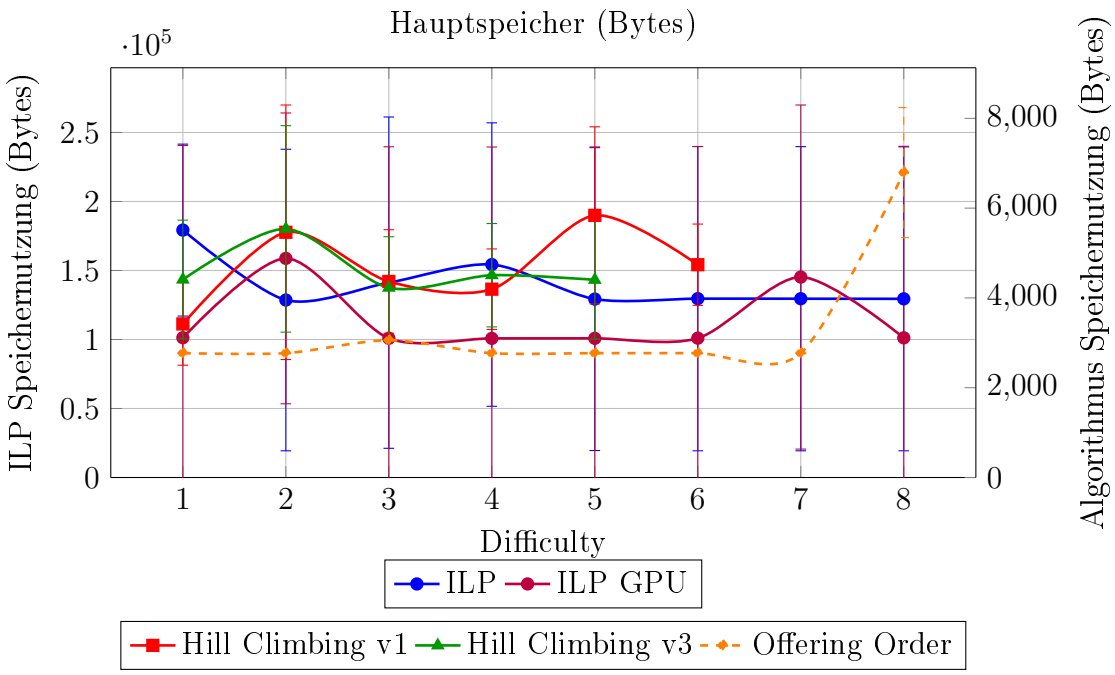



Hauptspeicher (Bytes)
·10^5
·10^4
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
0
2
4
6
8
0
0.2
0.4
0.6
0.8
1
1.2
0
1
2
3
4
5
6
7
8
9
10
11
12
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

5.14 Szenario 14: Keine Kurse Montags, Dienstags und Mittwochs






Hauptspeicher (Bytes)
$\cdot 10^5$
ILP Speichernutzung (Bytes)
Algorithmus Speichernutzung (Bytes)
2.5
2
1.5
1
0.5
0
8,000
6,000
4,000
2,000
0
1
2
3
4
5
6
7
8
Difficulty
ILP
ILP GPU
Hill Climbing v1
Hill Climbing v3
Offering Order

6 Diskussion

6.1 Vergleich der Ergebnisse mit Ursprungsarbeit?

6.2 Einschränkungen

7 Fazit

7.1 Verwandte Arbeiten

7.2 Anwendungsfälle und Weiterentwicklung

7.3 Schlussworte

References

[1] Vorlesungsverzeichnis der wirtschaftsuniversität wien. `https://vvz.wu.ac.at/`.

[2] Douglas G. Bonett. Design and Analysis of Replication Studies. *Organizational Research Methods*, 24(3):513–529, 3 2020.

[3] Mark J. Brandt, Hans IJzerman, Ap Dijksterhuis, Frank J. Farach, Jason Geller, Roger Giner-Sorolla, James A. Grange, Marco Perugini, Jeffrey R. Spies, and Anna Van 't Veer. The Replication Recipe: What makes for a convincing replication? *Journal of Experimental Social Psychology*, 50:217–224, 10 2013.

[4] Leonard E. Burman, W. Robert Reed, and James Alm. A Call for Replication Studies. *Public Finance Review*, 38(6):787–793, 10 2010.

[5] Michael W. Carter, Gilbert Laporte, and Sau Yan Lee. Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, 47(3):373–383, 3 1996.

[6] Mei Ching Chen, San Nah Sze, Say Leng Goh, Nasser R. Sabar, and Graham Kendall. A survey of university course timetabling problem: Perspectives, trends and opportunities. *IEEE Access*, 9:106515–106529, 2021.

[7] Peter Cowling, Graham Kendall, and Naimah Mohd Hussin. A survey and case study of practical examination timetabling problems. *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002*, 01 2002.

[8] M. Dostert, A. Politz, and H. Schmitz. A complexity analysis and an algorithmic approach to student sectioning in existing timetables. *Journal of Scheduling*, 19(3):285–293, 4 2015.

[9] R. Feldman and M. C. Golumbic. Optimization Algorithms for Student Scheduling via Constraint Satisfiability. *The Computer Journal*, 33(4):356–364, 4 1990.

[10] Wan Zuki Azman Wan Muhamad, Farah Adibah Adnan, Zainor Ridzuan Yahya, Ahmad Kadri Junoh, and Mohd Hafiz Zakaria. Solving university course timetabling problems using FET software. *AIP conference proceedings*, 2013:020052, 1 2018.

[11] Ahmad Muklason, Andrew J. Parkes, Ender Özcan, Barry McCollum, and Paul McMullan. Fairness in examination timetabling: Student preferences and extended formulations. *Applied Soft Computing*, 55:302–318, 1 2017.

[12] Tomáš Müller and Keith Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181(1):249–269, 3 2010.

[13] Hochschule Trier Trier University of Applied Sciences. Evaluation zu den Freiräumen im Stundenplan. 1 2024.

[14] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10):e1003285, 10 2013.

[15] A. Schaerf. A survey of Automated Timetabling. *Artificial Intelligence Review*, 13(2):87–127, 4 1999.

[16] Philipp Scheer. `https://github.com/philippscheer/bachelorarbeit/blob/main/models/ilp.py`.

[17] Philipp Scheer. `https://github.com/philippscheer/bachelorarbeit/blob/main/models/ilp_gpu.py`.

[18] Philipp Scheer. `https://github.com/philippscheer/bachelorarbeit/blob/main/models/hill_climbing_v1.py`.

[19] Philipp Scheer. `https://github.com/philippscheer/bachelorarbeit/blob/main/models/hill_climbing_v3.py`.

[20] Öh-wu studienplaner.