Algorithm Engineering for some Complex Practice Problems

Exact Algorithms, Heuristics and Hybrid Evolutionary Algorithms

Dipl.-Math. Volkmar Sauerland

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel
eingereicht im Jahr 2012

1. Gutachter:	Prof. Dr. Srivastav Christian-Albrechts-Universität zu Kiel
2. Gutachter:	PD Dr. Jäger University of Umea, Christian-Albrechts-Universität zu Kiel

Datum der mündlichen Prüfung: 3. September 2012

Zusammenfassung

Diese Arbeit befasst sich mit dem Entwurf von exakten und heuristischen Lösungsverfahren für schwere Optimierungsprobleme, die aus drei praktischen Anwendungen (*Tourenplanung*, *Lehrgangsplanung*, *Parameteroptimierung*) und einer klassischen kombinatorischen Fragestellung stammen.

Exakte Verfahren bekommen wir durch die Modellierung als mathematische Optimierungsprobleme und die Anwendung geeigneter Standardsoftware für deren Lösung. Für das Tourenplanungsproblem geben wir unter Ausnutzung der Problemstruktur zusätzlich ein angepasstes exaktes Verfahren an (inklusive Beweis der Exaktheit). Die Komplexität der behandelten Probleme verbietet allerdings das exakte Lösen großer Instanzen in adäquater Zeit.

Effiziente Heuristiken können wir zumeist durch geeignete Erweiterungen bekannter Verfahren für verwandte mathematische Probleme der vorliegenden Praxisaufgaben gewinnen. Zum Anderen haben sich evolutionäre Algorithmen sehr erfolgreich bei der Bewältigung vieler mathematisch schwerer Optimierungsprobleme gezeigt, sodass wir in unseren Experimenten für das jeweilige Problem geeignete EA-Rahmenwerke ermitteln. Eine wichtige Rolle wird dabei eine geeignete Hybridisierung mit den zuvor gewonnenen Heuristiken spielen.

Hinsichtlich Heuristiken sind Gütegarantien der erzeugten Lösungen in Relation zum globalen Optimum von theoretischem Interesse. Wir können für unsere Praxisaufgaben keine allgemeinen Gütegarantien im Voraus geben. Die beiden Planungsprobleme gestatten dieses beweisbar nicht. Allerdings können wir für gegebene Instanzen der Planungsprobleme untere Schranken für deren Minimalwert gewinnen, indem wir Lösungen von Relaxierungen der exakten mathematischen Modelle berechnen. Eine Verallgemeinerung unseres dritten Praxisproblems ist die parametrische nichtlineare Ausgleichsrechnung. Um für konkrete Instanzen dieses Typs untere Schranken zu beweisen, können wir unter bestimmten Voraussetzungen eine Technik benutzen, die auf einer parameterfreien Relaxierung beruht.

Abstract

This work deals with the design of exact algorithms and heuristics for complex optimization problems that origin from three practical applications in *tour planning*, *course scheduling* and *parameter optimization* and one classical combinatorial task.

We obtain exact algorithms by modeling our problems in terms of mathematical optimization problems and applying suitable standard software tools to solve this models. For the tour planning problem, we additionally provide an own exact algorithm (we give a prove of its exactness) that utilizes some structural properties. Because of the complexity of our problems, exact algorithms cannot solve large instances within adequate time.

Therefore, on the one hand, we derive efficient heuristics by adapting algorithms for similar mathematical problems in a suitable manner. On the other hand, evolutionary algorithms have shown to be successfully applicable to many hard mathematical problems. For that reason, we will experimentally determine appropriate EA frameworks. Regarding this, hybridization of the evolutionary operators with suitable problem specific heuristics will play an important role.

Concerning heuristics, guaranties on the quality relation of calculated solutions and the global optimum are of theoretical interest. For the practice problems considered in this work we can give no general advance guaranties. Such results are impossible for both the tour planning problem and the scheduling problem. But dealing with certain instances of both problems, lower bounds on their minimal value are given by relaxed solutions of the corresponding exact mathematical models. A generalization of our third practice problem is parameterized non-linear regression. Under certain circumstances, we can use a kind of parameter-free relaxation in order to prove lower bounds for given instances of this problem type.

"Properly speaking, such work is never finished;
one must declare it so when, according to time
and circumstances, one has done one's best".

Goethe (German):
„So eine Arbeit wird eigentlich nie fertig,
man muss sie für fertig erklären, wenn man nach Zeit
und Umständen das Möglichste getan hat“.

Preface

In industrial practice as well as in practical research tasks like "doing a job as cheap as possible" or "fitting model parameters to measured data" appear to have descriptions in terms of *mathematical optimization problems.* A very general mathematical optimization problem formulation is

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & x \in S, \end{array} \tag{0.1}$$

with *objective function* f from a set X to an ordered set (usually $\mathbb{R}$) and a subset $S \subseteq X$.

Optimization Methods

Practice problems lead to various specializations of (0.1) which are differently hard to solve and, thus, imply different demands on solution algorithms like obtaining

1. the very best solution to the problem,
2. approximative solutions within warranted limits related to the very best solution,
3. good approximative solutions after acceptable calculation time.

In Chapter 1 we give a brief overview on mathematical optimization problem types and their complexity along with a rough classification of optimization algorithms and some important general algorithmic techniques.

Topic of this Work

Our focus is on $\mathcal{NP}$-hard practice problems that can be modeled in terms of certain specification types of (0.1), namely (in three of four cases) by *integer linear programs* and (in one case) by a *non-linear, non-convex* but *smooth* optimization problem. Our algorithmic view covers demands 1. and 3. from above, i. e.,*exact algorithms* and efficient *heuristics* for the problems. Regarding heuristics, we distinct two algorithmic aspects. On the one hand, we deal with problem specific tailored algorithms. On the other hand, as effective application of *evolutionary algorithms (EA)* to several $\mathcal{NP}$-hard problems is documented in literature, we examine the effect of hybridizing suitable EA frameworks with our tailored algorithms.

Practice Problems

Concerning the above algorithmic approaches, Chapters 2 to 5 separately deal with one of our four practice problems. Here, we give a short introduction to these problems and our contributions.

TSP with Multiple Time Windows

This task was posed by an industrial cooperation with the FLS GmbH, Heikendorf, a leading company for tour planning software in Germany. The cooperation was founded by the program Hochschule-Wirtschaft-Transfer (HWT) of the Innovationsstiftung Schleswig-Holstein (ISH) lasting from 2005 until 2006. The matter is to plan multi-day trips for single employees of companies providing field service. Here, customers provide certain opening time windows that must be met. Further constraints are given by working rules, hotel options and penalty fees for omitted customers.

We provide a mixed integer linear programming model (MILP) which we solve by two exact algorithms. On the one hand, we apply the standard Cplex ILP solver to our model. On the other hand, we developed our own branch and bound algorithm that accelerates search by utilizing some properties of the problem. We also prove the exactness of our branch and bound approach. In order to provide an efficient (inexact) algorithm we developed a problem specific random insert heuristic. The solutions of the heuristic can be checked by the exact methods and are observed to be globally optimal for most of the small instances. Finally, we provide an EA framework that operates on the arguments of the heuristic.

Ocean Model Parameter Optimization

Within research area "Oceanic CO_2 Uptake" of "The Future Ocean" cluster of excellence we were concerned with parameter optimization of a biogeochemical ocean model. This so called *NPZD* model simulates the circulation of nitrogen in a vertical water column. It supposes nitrogen to occur in four different states (which are eponym for the model) in the ocean: dissolved in water as inorganic nitrogen (N), within phytoplankton (P), within zooplankton (Z) and within detritus (D), i. e., dead organic particles. The parameters that have to be optimized belong to functions describing the change of nitrogen states and vertical nitrogen fluctuation within partial differential equations. The goal is to adapt those parameters in order to best fit accordant measured data.

Together with the research group *Algorithmic Optimal Control - CO_2 Uptake of the Ocean* we developed an optimization framework that hybridizes genetic operators with deterministic gradient based search and that may also be applied to other parameter optimization problems [RSS^+10]. Our experiments with that framework give more evidence for conjectures of marine biologists about the need to revise/extend the ocean model. Our second contribution is the introduction of a method to determine lower error bounds for a generalization of our problem: non-linear regression. The method utilizes smoothness properties of the model function and may confirm its insufficiency, if high data resolution is given and there exist processes of high frequency/amplitude that are not incorporated into the model function.

Course Scheduling

Like TSP with Multiple Time Windows this problem comes from industrial practice and was also founded by the HWT program of the ISH (from 11/2009 to 2/2011). Our cooperation partner is the MINT Software Systems GmbH, Kiel, Germany. MINT provides course scheduling software for personal training institutions in aviation industries. Concerning long term planning (up to two years), there may be up to several hundred courses to be held by several hundred instructors for several thousand trainees. High planning complexity is given by various kinds of constraints, concerning curriculums of courses, qualifications and working rules of human resources, capacities of rooms, etc.

Within the limits of our project we focused on two approaches. On the one hand, we provide a quite comprehensive time indexed ILP model of the scheduling problem. On the other hand, a sophisticated *schedule generation scheme (SGS)* that considers the relationships between the MINT constraints was developed as a planning heuristic and also embedded as encoding function within a hybrid EA framework. The development of the SGS and its implementation along with the EA framework is the work of Torben Rabe. Details of the SGS will be presented within his Diploma Thesis [Rab12]. Here, we will only give a rough description of its working principles. Similar to the TSPTW problem, we are able to check the validity of heuristic solutions with the ILP and vice versa. Exact solutions are calculable for very small instances only, since the ILP formulation requires many variables and constraints. However, precalculations should reduce the size of the ILP and parts of it are intended to be utilized within LP-based heuristics in future.

Discrepancy of Arithmetic Progressions

This problem does not origin from industrial practice nor practical research but is a classical research object in combinatorics. Arithmetic progressions are finite or infinite sequences of numbers with constant difference between each two successive members. Considering the set $[n]$ of the first n positive integers, one may ask for a partition $[n] = N_1 \dot\cup N_2$ such that all arithmetic progressions in $[n]$ are quite balanced between N_1 and N_2. The balance measure of most relevance in research will be formally introduced in Chapter 5. It is remarkable that, in terms of this measure, the exact asymptotic order (w.r.t. n) of the best partition is known by now, but an efficient algorithm for computing such a partition is unknown.

Using an ILP formulation of the problem, we are able to calculate exact solutions for two-digit n, only. As we will show, the ILP matrix is dense and of the size n times $n^2 \log n$. We provide a hybrid framework with the option to either use a classical EA or a so called quantum inspired EA. Hybridization is done by an encoding function that utilizes the features of a problem method with the best theoretical performance guaranty. We prove that our framework guarantees the same exact asymptotic ratio. But the EA solutions have minimum objectives within $\frac{4}{5}$ of the objectives of the base algorithm.

Acknowledgements

I thank my doctoral adviser Prof. Dr. Srivastav for providing me with the research opportunities that led to this thesis and for his support throughout my studies.

I thank C. Patvardhan for acquainting us with the field of evolutionary algorithms and his suggestions concerning this matter during his research visits in Kiel.

I thank Andreas Baltz, Mourad El Ouali and Gerold Jäger for common research on the traveling salesperson problem with time windows.

I thank Andreas Oschlies, Thomas Slawig and both their research groups for the joint work and discussions concerning the parameter optimization problem of the biogeochemical ocean model.

I thank Torben Rabe for common research on the MINT scheduling problem.

I thank Werner Lehmann from the MINT company for introducing us to the complex issues of the MINT scheduling problem, for several discussions and for providing the MINT JAVA environment as well as a couple of problem instances that were necessary for our experiments.

I thank Nils Hebbinghaus for his hints concerning the theoretical bounds of Sárközy's algorithm.

I thank Lasse Kliemann for proofreading the chapter on discrepancy of arithmetic progressions.

I thank the initiators and organizers of the Kiel Computer Science Series (KCSS) for the good publishing opportunity it provides.

I thank both our cooperation companies, MINT and FLS, for their support during the initiation and implementation of our projects.

I thank the Innovationsstiftung Schleswig-Holstein for the financial support of both industrial corporations.

I thank my parents for their love and support.

And I thank everyone else who supported me during my studies, research, and writing of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Problem Types, Complexity and Algorithms

In this chapter we give a rough overview on mathematical optimization problem types, especially those in terms of which we will formulate our practice problems of Chapters 2, 3, 4 and 5. Since all our practice problems belong to the class of $\mathcal{NP}$-hard problems, we briefly discuss the topic of complexity classes as well as some different algorithmic paradigms to handle hard problems. We also introduce some general algorithmic techniques that are applicable to our kinds of problems.

1.1 Problem Types

Considering a set X and an (objective) function f from X to $\mathbb{R}$, the general mathematical optimization problem

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & x \in S, \end{array} \tag{1.1}$$

with *feasible* set $S \subseteq X$, can be categorized by

1. the argument set X,
2. the function type of f,
3. the (description of the) feasible set S.

Usual argument sets are $\mathbb{R}^n$, $\mathbb{Z}^n$, $\{0,1\}^n$ (referring to *continuous optimization*, *integer optimization* and *combinatorial optimization*, respectively) or products of these sets. Further frequent argument sets for combinatorial problems are power sets and sets of permutations. Some important properties of f are *smoothness*, *convexity* and *linearity*. If the feasible set S is not identical to the argument set X the problem is said to be *restricted*, otherwise it is said to be *unrestricted*. The feasible set is usually described in terms of equality and/or inequality constraints as follows

$$S = \left\{ x \in X \;\middle|\; \begin{array}{l} g(x) = 0, \\ h(x) \leqslant 0 \end{array} \right\}$$

with constraint functions $g : X \mapsto \mathbb{R}^{m_1}$ and $h : X \mapsto \mathbb{R}^{m_2}$, $m_1, m_2 \in \mathbb{N}_0$. Similar to f, the constraint functions provide further problem categorizations by function type.

A wide amount of work considers the situation where $X = \mathbb{R}^n$ and f, g and h are smooth (but non-linear in general), deriving optimality conditions for solutions and iterative solution algorithms utilizing those conditions. Monographs like [GMW81][Kos93][OR00][BV04] are dedicated to this field. The parameter optimization problem of Chapter 3 is a special variant of this situation, namely a *non-linear least-squares problem*. There, the objective function has the form $f = \sum_{i=1}^{N} (F_i)^2$ and the constraint functions are given by box constraints on the parameters (arguments of f).

Every other problem treated in this work is of combinatorial nature (having discrete arguments) and allows a description by an *integer linear program (ILP)* or *mixed integer linear program (MILP)*. An ILP is an optimization problem with argument set $X \subseteq \mathbb{Z}^n$ and linear objective and constraints

$$\begin{aligned} f(x) &= c^T x, \\ g(x) &= Ax - b, \\ h(x) &= \widetilde{A}x - \widetilde{b}, \end{aligned}$$

$c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m_1 \times n}$, $b \in \mathbb{R}^{m_1}$, $\widetilde{A} \in \mathbb{R}^{m_2 \times n}$, $\widetilde{b} \in \mathbb{R}^{m_2}$, compactly written as

$$\begin{array}{ll} \text{minimize} & c^T x, \\ \text{subject to} & Ax = b, \\ & \widetilde{A}x \leqslant \widetilde{b}, \\ & x \in X. \end{array} \tag{1.2}$$

If some but not all arguments are integral, i. e.,

$$X = \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}$$

with $n_1, n_2 \in \mathbb{N}_0$, $n := n_1 + n_2 \neq n_2$, (1.2) defines a MILP. Modeling combinatorial problems by ILPs in accordance with exact algorithmic approaches to tackle ILPs came into the focus of interest in the middle of the 20th century (see Section 1.3).

1.2 Hard Problems

Our practice problems belong to the algorithmic complexity class of $\mathcal{NP}$-hard problems. We give a brief introduction to complexity (classes), here. Details about this topic are covered by pertinent monographs like [VL90][PS98][Weg05].

1.2.1 Complexity of Algorithms

The aim to solve problems efficiently exists since ancient times. The *Sieve of Eratosthenes* (determining all prime numbers up to a given natural number) and *Euclid's Algorithm* (calculating the greatest common divisor of two integers) are examples to that effect. With respect to a certain instance of a problem, the computational effort (efficiency, performance) of an algorithm can be measured by counting the required number of arithmetic operations or time units. Regarding computer systems, the required memory size is also an important factor.

1.1 Remark. Talking about an algorithm and its effort, we intuitively think about a procedure that enables us to evaluate a function by performing a number of elementary calculations on both the input and recorded interim results. This procedure might either be done by humans (using paper and pencil) or by computers. One of the first theoretical models to this approach is the *Turing machine* [Tur36] (1936). The development of complexity theory formally relies on it (also see [FH03]).

Dealing with certain problem instances only, comparison of two algorithms intended to solve the same problem may lead to inconsistent performance results. For that reason, the main interest concerns the asymptotic performance behavior of algorithms with respect to some characteristic parameter of the instance size.

1.2 Definition. Let $\mathcal{A}$ be an algorithm for a problem P and $\mathcal{I}_P$ the set of all instances of P. Let $t_{P,\mathcal{A}} : \mathcal{I}_P \to \mathbb{N}$ be a function that assigns every problem instance of P to the computational effort $\mathcal{A}$ requires to solve it. Let $|\cdot|_P : \mathcal{I}_P \to \mathbb{N}$ be some measure of the instance size. The *worst case complexity* $T_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ of $\mathcal{A}$ is defined by

$$T_{\mathcal{A}}(n) = \max\{t_{P,\mathcal{A}}(I) \mid I \in \mathcal{I},\ |I|_P = n\}. \tag{1.3}$$

Asymptotic complexity bounds on algorithms are commonly expressed by *Landau Symbol* notation, e. g.,

$$T_{\mathcal{A}} \in \mathcal{O}(f), \tag{1.4}$$

meaning that there is a positive constant C and an $n_0 \in N$ such that $T_{\mathcal{A}}(n) \leqslant Cf(n)$ for all $n \geqslant n_0$. An efficient algorithm $\mathcal{A}$ satisfies (1.4) for some function f that increases slowly with n, e. g., $f(n) \in \{\log(n), n, n^2, n^3\}$ referring to *logarithmic(-time/space) algorithm*, *linear(-time/space) algorithm*, *quadratic(-time/space) algorithm* and *cubic(-time/space) algorithm*, respectively. From the practical point of view, high powers of n do not describe asymptotic performance of efficient algorithms any more. But (looking ahead to subsection 1.2.2) the theoretically essential term of efficiency is *polynomial(-time/space) algorithm*, meaning that an algorithm $\mathcal{A}$ satisfies $T_{\mathcal{A}} \in \mathcal{O}(n^k)$ for some $k \in \mathbb{N}$.

In addition to the worst case complexity definition, one sometimes considers the *average case complexity.* We can replace the term "max" in (1.3) by the term "expectation" to define it. Average case complexity estimations require knowledge or assumptions on the distribution of the problem instances.

1.2.2 Complexity of Problems

Efficient solution algorithms have been found for many problems, but until the early computer age an increasing number of problems was recognized to resist every endeavor concerning polynomial-time solution algorithms. Popular examples are the *traveling salesman problem* and *scheduling problems*, variants of which we consider in Chapter 2 and Chapter 4. The aim to fix similarities of those "hard" problems in order to obtain more evidence that they are not solvable in polynomial-time led to the problem classes $\mathcal{P}$, $\mathcal{NP}$

and $\mathcal{NPC}$. We give informal descriptions of these concepts, here.

Problems of the form (1.1) are optimization problems. Corresponding *decision problems* concern the question if there is a feasible solution such that the objective function is bounded by a given value. The class $\mathcal{P}$ consists of all decision problems for which a polynomial-time solution algorithm exists. A probably easier task is to verify, if some suggested solution of a decision problem is an actual solution. An algorithm that does this task is called *verification algorithm.* The class $\mathcal{NP}$ consists of all decision problems for which a polynomial-time verification algorithm exists. In fact, most of the "hard" problems mentioned above are in $\mathcal{NP}$ (but not known to be in $\mathcal{P}$).

Another issue is the hardness of a decision problem P in relation to a second decision problem Q. An algorithm $\mathcal{A}$ for P is called *Karp reduction* from P to Q if

- $\mathcal{A}$ is a polynomial-time algorithm,
- for every instance I of P the output $\mathcal{A}(I)$ of $\mathcal{A}$ is an instance of Q,
- for every instance I of P, the answer to I is true, if and only if the answer to $\mathcal{A}(I)$ is true.

The problem P is said to be *Karp reducible* to Q, denoted by $P \leqslant Q$, if a Karp reduction from P to Q exists. The properties of a Karp reduction immediately yield that P is in $\mathcal{P}$ (resp. $\mathcal{NP}$) if Q is in $\mathcal{P}$ (resp. $\mathcal{NP}$) and $P \leqslant Q$. We also say that P is not harder than Q, if $P \leqslant Q$.

The (virtually contrariwise) informal issue that a decision problem is at least as hard as the hardest problems in $\mathcal{NP}$ is formalized by the terms $\mathcal{NP}$-hardness and $\mathcal{NP}$-completeness. A decision problem Q is said to be $\mathcal{NP}$-hard if $P \leqslant Q$ holds for every problem P in $\mathcal{NP}$. A problem in $\mathcal{NP}$ that is also $\mathcal{NP}$-hard is said to be $\mathcal{NP}$-complete. The class $\mathcal{NPC}$ consists of all $\mathcal{NP}$-complete problems. Cook [Coo71] (1971) and Levin [Lev73] (1973) independently proved that an $\mathcal{NP}$-complete problem indeed exists. Karp [Kar75] (1972) showed 20 further problems to be in $\mathcal{NPC}$ by reducing the *Boolean satisfiability problem (SAT)* (the problem, Cook proofed to be in $\mathcal{NPC}$) to them. An increasing number of known $\mathcal{NP}$-complete problems made it easier to prove even more problems to be in $\mathcal{NPC}$ by Karp reduction. In fact, most of the problems which were suspected

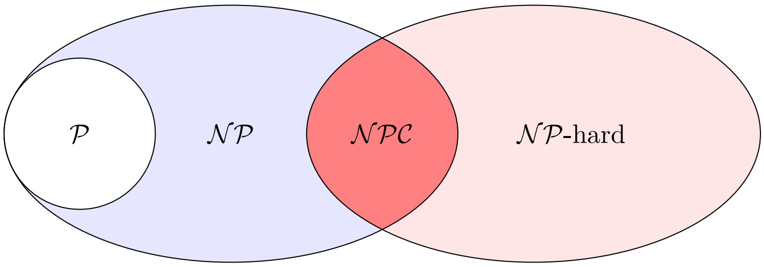


Figure 1.1. Complexity classes $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{NPC}$, provided that $\mathcal{P} \neq \mathcal{NP}$.

to have no polynomial-time solution algorithm turned out to be in $\mathcal{NPC}$. This implies that a polynomial-time solution algorithm for any of these problems would indirectly solve every other problem in $\mathcal{NP}$ and prove $\mathcal{P} = \mathcal{NP}$. Thus, the conjecture that none of these problems is solvable in polynomial time comes down to the still open conjecture that $\mathcal{P} \neq \mathcal{NP}$.

1.2.3 $\mathcal{NP}$-hard Types

Integer linear programs are $\mathcal{NP}$-hard in general [GJ78]. The same holds for non-convex quadratic programs [Sah74].

1.3 Exact Algorithms

Despite the fact that a problem is $\mathcal{NP}$-hard there is still the prospect to find algorithms that are more efficient than others and able to solve quite large instances to optimality in a satisfying amount of time. Most popular example to that effect is the *traveling salesman problem (TSP)* a variation of which we will consider in Chapter 2. An instance of the TSP consists of a number of cities and their pairwise distances. The optimization version of TSP asks for a shortest tour that visits every city exactly once. Its decision version is one of the 20 problems Cook proved to be $\mathcal{NP}$-complete. The TSP was also benchmark problem for many inexact algorithmic approaches, some of which are utilized in this work and discussed in the subsequent sections.

1.3.1 Brute Force

Brute force means full enumeration of all possible solutions and is the most simple way to deal with finite optimization problems. The effort of brute force corresponds to the size of the solution space (feasible set) which grows exponential with the input size already for many easy combinatorial problems. For example, the TSP on n cities has $(n-1)!/2$ feasible solutions, a number which exceeds the number of atoms in the cosmos, if $n > 60$.

1.3.2 Relaxation based Algorithms

A relaxation of a problem emerges if its feasible set is enlarged. Under certain circumstances relaxations are comparatively easy to solve and may be utilized by algorithms for the original problem. This idea plays an important role in the context of (M)ILP models. Here, enlargement of the feasible set means omission of the integrality conditions of the (M)ILP resulting in a corresponding linear program (LP) which is called LP-relaxation of the M(ILP). While (M)ILP models in general belong to the $\mathcal{NP}$-hard problems, it has been shown that LPs are polynomial-time solvable. The first proof of this fact was given by Khachiyan [Kha79] (1979) using the *ellipsoid algorithm.* The method described by Khachiyan was not practically efficient. On the contrary, variations of Dantzig's *simplex algorithm* [Dan63] for linear programming have shown to be very efficient in average but have exponential worst case performance. Karmarkar's algorithm [Kar84] was the first *interior point method* that provided both theoretical and practical efficiency.

Cutting Plane Method

The treatment of a 49 city TSP instance by Dantzig, Fulkerson and Johnson [DFJ54] (containing Washington D.C. and all local capitals of the U.S.) was a breakthrough for the solution of TSP problems. They started with an ILP model which contained some easy necessary conditions, solved its LP-relaxation, identified further necessary conditions that were violated by the solution and added them in terms of new linear inequalities to the LP. This procedure was repeated until the LP-solution represented an actual tour which was implicitly optimal. It was the invention of *cutting* *planes* and the first successful application of integer linear programming as a tool for solving combinatorial problems. Dealing with integer linear programs, the general idea of the cutting plane method is to successively add linear inequalities (cutting planes) to its LP-relaxation such that the current solution but no integral solution is excluded from the feasible set. If the current LP-solution is integral, it must be optimal to the original ILP because the current LP still describes a superset of the feasible set. Figure 1.2a) shows a 2-dimensional example. Gomory [Gom58] gave the first generally applicable method to produce such cutting planes (*Gomory cuts*). But despite their success in the problem specific usage, cutting planes became no adequate general purpose method until being combined with *branch and bound.*

Branch and Bound

The branch and bound (*BnB*) method is intended to accelerate the search for the optimal solution x^* of an integer optimization problem by excluding parts of its feasible set. It was first introduced by Land and Doig [LD60]. Dakin [Dak65] described a branch and bound algorithm that is easier to implement. The essence of the procedure is as follows. The feasible set S is successively split (*branched*) into subsets S_i, $i = 1, \dots,$ defining a search tree with vertices S_i. The subsets S_i are relaxed to sets R_i on which an optimal solution $x^{(i)}$ is efficiently computable. The objective value $\alpha^{(i)}$ of $x^{(i)}$ is a lower *bound* for the best solution in S_i and may be improved to the maximum corresponding bound of the subsets of S_i as soon as all their relaxations have been solved. If $x^{(i)}$ is integral itself, $\alpha^{(i)}$ is also an upper *bound* on the value of x^*. A subset S_i can be excluded from the search space if its lower bound is found to be greater than the minimal, so far, upper bound on the value of x^*.

Branching is often done at fractional components of the solution of the current relaxation, say $x_i = \beta \in \mathbb{R}\backslash\mathbb{Z}$. Then, one subset is forced to contain every solution with $x_i \leqslant \lfloor\beta\rfloor$ and the other subset is forced to contain every solution with $x_i \geqslant \lceil\beta\rceil$. Figure 1.2b) illustrates branching at an ILP variable in the 2-dimensional case. In practice, the performance of a branch and bound strategy significantly depends on the order in which the branching variables are considered and the corresponding sub-branches are searched.

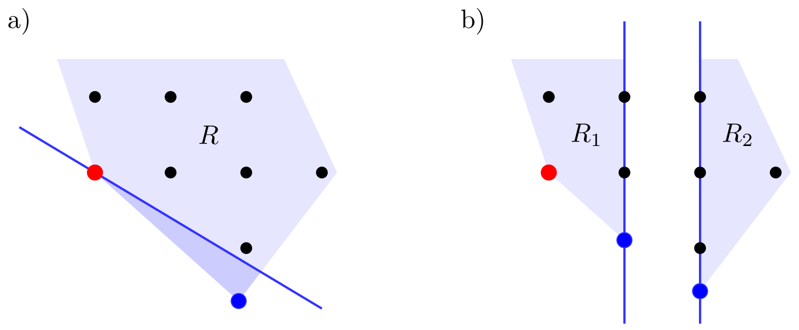


Figure 1.2. a) Cutting plane application to the LP relaxation of an ILP. The polyhedron R is the feasible set of the LP relaxation. It contains the optimum of the ILP (red dot) and some further ILP solutions (black dots). The LP optimum (blue dot) gets separated from the ILP solutions by the cutting plane (blue line). b) Branching at a variable with fractional value in the LP optimum splits the feasible set of the ILP, yielding new relaxations R_1 and R_2 and new fractional optima (blue dots).

Branch and Cut

Branch and cut (BnC) combines branch and bound with the cutting plane method and is by now one state of the art technique for the solution of (mixed) integer linear programs. Branch and cut first tightens the initial LP-relaxation by iterative addition of cutting planes, making the subsequent branch and bound procedure more efficient. Further cutting planes may also be found and applied to subproblems within the branch and bound procedure.

Standard solvers like CPLEX (which we apply to solve instances of our (M)ILP models), XPRESS and others use branch and cut algorithms resting upon simplex type LP solvers.

The success of branch and cut was again driven by research on the TSP problem (see e. g., [CP80]). According to [JLN$^+$10], a first fully automatical algorithm was provided by Hong [Hon72] and (among others) advanced by Grötschel and Padberg bringing in the benefit of their polyhedral research (see e. g., [GP85]).

1.4 Heuristics

Complexity theory gives strong evidence that no $\mathcal{NP}$-complete problem is solvable in polynomial time. In particular, no accordant optimization problem is solvable to optimality in polynomial time. Yet, dropping the hope to find exact solutions for large instances of such optimization problems, the "compensation paradigm" is to efficiently find reasonable good solutions for them. This is what heuristics are supposed to achieve.

1.4.1 Approximative Algorithms

Having no optimality proof for a heuristic problem solution, it is interesting to know something comforting about its value in relation to the optimum. Algorithms that permit some kind of guaranty on this relationship belong to the category of *approximative algorithms.* The book of Jansen and Margraf [JM08] gives a wide survey of this topic. A particularly nice approximative algorithm type is the *fully polynomial time approximation scheme (FPTAS).* For every $\varepsilon > 0$, an FPTAS provides approximations of the optimal solution value within a factor of $1 + \varepsilon$ in a time that is polynomial in both the instance size and $1/\varepsilon$. Weaker are "normal" *polynomial time approximation schemes (PTAS)*, which provide the same but are not polynomial in $1/\varepsilon$, and polynomial algorithms that provide a constant factor approximation.

The intensively researched example problem, TSP, has no approximation algorithm with constant approximation ratio in general. In the metric case, i. e., if distances satisfy the Δ-inequality, TSP permits 1.5 approximation (*Christofides Algorithm*) [Chr76]. For the Euclidian TSP there is even a PTAS (*Arora Algorithm*) [Aro98]. The TSP variation DTSP is defined by imposing visiting deadlines to the cities (measuring distances in terms of time units). It was shown in 2007 that, even in the metric case, no polynomial constant factor approximation exists for DTSP [BHKK07].

For the problems considered in this work we can give none of the advance estimations introduced above. Since DTSP is reducible to the problems considered in Chapter 2 and Chapter 4 such results are impossible, there. But at least we can obtain computational estimations for given instances via model relaxations.

1.4.2 Relaxation based Heuristics

As discussed in subsection 1.3.2, exact algorithms like branch and cut for integer linear programming utilize relaxations of (sub)problems to accelerate the search. Polynomial time solvable relaxations may also be utilized by efficient heuristics.

First of all, an exact algorithm like branch and bound (branch and cut) can serve as a heuristic, if the search is stopped after some satisfying integral solution has been found. As there will also be a best lower bound due to the relaxations solved until termination, one additionally obtains an upper bound on the approximation ratio.

Randomized Rounding

Suppose we have a binary optimization problem (i. e., the argument set is $\{0,1\}^n$) and a vector $p \in [0,1]^n$, with $p = (p_1, \cdots, p_n)$. We wish to generate a binary solution $x = (x_1, \cdots, x_n) \in \{0,1\}^n$ guided by p. We can use naive rounding and set for $i = 1, \cdots, n$

$$x_i = \begin{cases} 1, & \text{if } p_i > 0.5 \\ 0, & \text{if } p_i \leqslant 0.5 \end{cases}$$

Such a rounding is not flexible, and applied to constrained optimization problems leads usually to infeasible solutions and bad objective values. Raghavan and Thompson [RT87] introduced a randomized version:

```
Randomized Rounding (RR($p$))
For $i = 1, \cdots, n$ independently set
```

$$x_i = \begin{cases} 1, & \text{with probability } p_i \\ 0, & \text{with probability } 1 - p_i \end{cases}$$

This is nothing else than a sequence of n independent Bernoulli trials X_i (or n flips of a biased coin), where for all $i = 1, \cdots, n$

$$\mathbb{P}[X_i = 1] = p_i \text{ and } \mathbb{P}[X_i = 0] = 1 - p_i.$$

The corresponding probability distribution on $\{0,1\}^n$ is

$$\mathbb{P}(x) = \prod_{\substack{i\in[n],\\ x_i=1}} p_i \prod_{\substack{i\in[n],\\ x_i=0}} (1-p_i) \text{ for } x \in \{0,1\}^n.$$

Such a rounding scheme is much more flexible compared to the naive rounding scheme, it is able to get out from local optima, and most important, can be analyzed with probabilistic methods, like large deviation inequalities of Chernoff, Hoeffding and others [RT87] [Sri01b].

1.4.3 Certain Heuristic Difficulties

For continuous optimization problems as well as for combinatorial optimization problems, well suited algorithms have been developed. Classical iterative methods are suited for continuous optimization problems. Those methods grant fast convergence to a local optimum within its neighborhood, if derivations of the objective function f (and optionally imposed inequality constraint functions g_i, $i \in [m_1]$ and equality constraint functions h_j, $j \in [m_2]$), exist. The good-natured case is given by convexity of both the objective function and the feasible set S (convexity of the g_i and affinity of the h_j) since this additionally grants every local optimum to be a global optimum. On the other hand, presence of multiple local optima in a continuous problem will make classical gradient-based methods likely to get stuck at one (maybe bad) of these, unless a suitable starting point is known. Figure 1.3 shows example objective function types having multiple local optima for the univariate case.

Concerning combinatorial optimization, there are tailored algorithms for every problem. The good-natured combinatorial problems are those in $\mathcal{P}$. Tailored algorithms for $\mathcal{NP}$-hard combinatorial problems also behave locally optimal, e. g., in the sense that

(a) every interim partial solution of a constructive algorithm is an optimal extension of its predecessor,

(b) the final solution is a local optimum w.r.t. a distance measure implied by a local search strategy.

Concerning the TSP, examples for *(a)* and *(b)* are *insert heuristics* and *k-opt heuristics*, respectively.

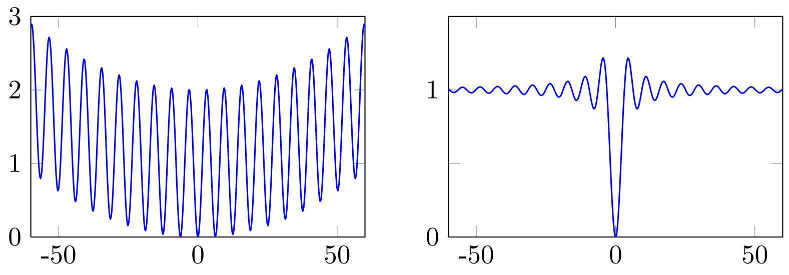


Figure 1.3. Univariate Griewank function (left) and modified sinc function (right)

1.3 Example. Let π be a permutation of n. For a TSP instance with n cities, insertion w.r.t. π starts with the unique tour $T_\pi^{(2)}$ through the cities $\pi(1)$ and $\pi(2)$ and subsequently builds $T_\pi^{(i)}$, $i = 3, \dots, n$, by inserting $\pi(i)$ at that position into $T_\pi^{(i-1)}$ that causes the cheapest loop way. In other words, for $i = 3, \dots, n$, the interim solution $T_\pi^{(i)}$ is the optimal extension of $T_\pi^{(i-1)}$ by $\pi(i)$. There are different strategies to build the permutation π (either in advance or during tour construction) that yield different insert heuristic variations like *nearest insertion*, *farthest insertion* or *random insertion*.

1.4 Example. One step of the local search heuristic k-opt tries to improve a given tour T by removing k edges and inserting other k edges such that the result is a new but cheaper tour T', where T' is chosen to be the best of all possibilities to change k edges. The step is repeated until it does not change the actual tour any more. Defining the distance $\delta(T, T')$ of two tours T and T' to be the number of edges that are not contained in both tours, a k-opt result is optimal in the k-neighborhood w.r.t. δ. The strategy origins from the idea to remove crossing edges (case $k = 2$) of tours in the Euclidian plane.

A quite opposite behavior to the above heuristic approaches is given by entirely random algorithms which draw samples from the search space under some probability distribution. Clearly, the absence of any convergence property prevents from getting trapped in a bad region of the search space S. However, for $\alpha \in \mathbb{R}$ the probability of the *sublevel set* $S_f(\alpha) := \{x \in S \mid f(x) \leqslant \alpha\}$ can be very small even if the global optimum is quite better than α.

1.5 Example. The n-dimensional version of the modified *sinc function* shown at the right hand side of Figure 1.3 is given by

$$\operatorname{sinc}(x) = 1 - \prod_{x_i \neq 0} \frac{\sin(x_i)}{x_i}.$$

Its global minimum is $\operatorname{sinc}(0) = 0$ (since the empty product is equal to 1). For every feasible set $S \subseteq \mathbb{R}^n$ it holds that $S_f(0.5) \subseteq [-2,2]^n$. If we choose $a \in R_{>1}$ and set $S = [-2a, 2a]^n$, the ratio of both volumes $\lambda(S_f(0.5))$ and $\lambda(S)$ is bounded by a^{-n}. Thus, the probability to sample an $x \in S_f(0.5)$ under uniform distribution shrinks exponentially with the problem dimension n.

1.5 Evolutionary Algorithms

Briefly speaking, evolutionary algorithms are randomized meta heuristics that simulate the mechanisms of natural evolution. Their origin goes back at least to the 1950th. In the 1960th and 1970th, significant research was carried out under the terms *evolution strategy (ES)* by Bienert, Rechenberg [Rec73] and Schwefel [Sch81], *evolutionary programming (EP)* by Fogel et al. [FOW66] and *genetic algorithm* by Holland [Hol75]. The three fields coexisted independently until the first *Parallel Problem Solving from Nature (PPSN) workshop* joined them in 1990. The collective terms *evolutionary computation (EC)* and *evolutionary algorithm (EA)* have been found for the common research field and its associated algorithms, respectively.

It has been found that computer code conversions of evolutionary mechanisms can be instrumental in the solution of several optimization tasks for which tailored heuristics as well as simple randomized heuristics face difficulties like those described in subsection 1.4.3. There, either a wide spread search of the solution space or fast convergence to (local) optimal solutions was not supported. In the context of evolutionary algorithms, these properties are usually termed *exploration* and *exploitation*, respectively. A problem specific EA is intended to strike a balance between both properties.

1.5.1 Classical Evolutionary Algorithm Framework

There are some general common properties of classical evolutionary algorithms that were inspired by natural evolution. An EA iteratively processes an initially generated finite (small) subset P of possible solutions to a problem. Using the terminology from natural evolution, a solution and the set P are called *individual* and *population*, respectively. As individuals are vectors of some fix dimension n, one component of an individual is called a *gene*. Further evolutionary terms are *locus* for the index of a gene and *allele* for its value.

One EA iteration is also designated as *generation* and includes the following steps. At first, a subset $P' \subseteq P$ is chosen from which a set P'' of new solutions is obtained due to the application of *variation operators*. A variation operator either performs random changes to a single individual or combines sets of (usually two) individuals to new ones. Those variation operations are referred to *mutation* and *recombination*, respectively. In accordance to natural evolution, if recombination is incorporated, the set P' is called *mating pool* and the individuals in P'' are called *offsprings*. The set P' can either be equal to P or assembled by random element *selection*. Using random selection, the probabilities can (but do not need to) depend on the objective values of the individuals in order to bring in some *selective pressure*. In the EA context, the objective value $f(x)$ of a solution x is also called *fitness*, supposing that f has to be maximized. Note, that every minimization problem can be transformed into a maximization problem, e. g., by using $-f$ instead of f. The set P is finally updated to $P \cup P''$ and usually resized to its old cardinality by a second fitness dependent selection (*survival of the fittest*). Figure 1.4 illustrates the generic EA procedure.

1.5.2 Encoding

Above, we said that an EA operates on the elements of the solution space of an optimization problem. This set may be the feasible set S of the actual mathematical optimization problem on hand. In other cases, the evolutionary operations can take place on a different, possibly easier structured set $\mathcal{G}$ which is mapped to the feasible set S of the actual problem by a so called *encoding function* γ. This means, that the EA optimizes a modified problem described by the set $\mathcal{G}$ and the induced fitness function $F = f \circ \gamma$. In this

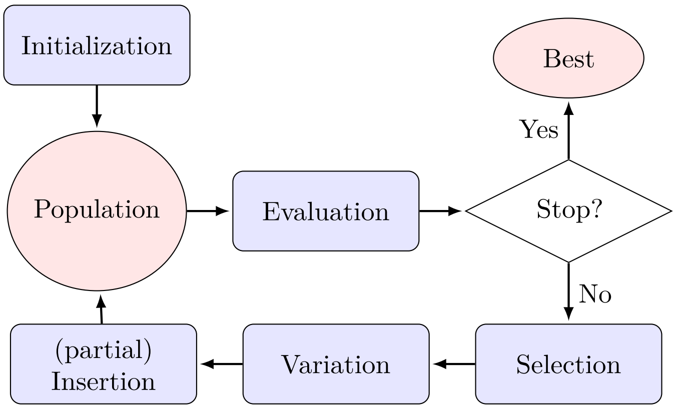


Figure 1.4. Schematic of a Classical EA

context, an individual $x \in \mathcal{G}$ and the corresponding solution $\gamma(x)$ of the actual problem are called *genotype* and *phenotype*, respectively. The standard example for encoding is genetic algorithms that use binary representations for the arguments of a continuous optimization problem, e. g., mapping $\mathcal{G} = \{0,1\}^{nl}$ to $S \subseteq \mathbb{R}^n$. But encoding can also be used to *hybridize* an EA by the incorporation of problem specific heuristics into γ. In the latter context, the encoding function is also said to be a *solution generation scheme*.

1.5.3 Selection

Many selection operators have been proposed for the choice of the mating pool and the integration of the offsprings, respectively. The aim to yield faster convergence toward good solutions is the motivation to bring in some amount of selective pressure by giving fit individuals a higher probability to breed than unfit ones. Very popular selection operator examples for the mating pool are *roulette-wheel selection (RWS)* and *tournament selection*, respectively. Roulette-wheel selection assigns each individual a selection probability that is proportional to either its fitness or some measure that is deduced from the fitness values of all individuals. The probabilities can be imagined as accordingly large segments of a roulette-wheel that is spun to realize the selection of one individual. Often, a ranking based valuation is used for an individual, i. e., the selection probability depends on its rank with respect to its fitness (the fittest individual has rank 1 and the most unfit individual has rank $|P|$). We will use ranking based roulette-wheel selection, choosing a selection probability of an individual $i \in P$ to be $2\frac{|P|+1-\text{rank}(i)}{|P|(|P|+1)}$. This refers to linear ranking based roulette-wheel selection with selective pressure 2 as described in [Ree03].

While each roulette-wheel selection uses global information, one tournament selection performs a local evaluation. The basic version of tournament selection uniformly draws a subset of $k \in [|P|]$ individuals for a competition which is won by the best of them. The tournament size parameter k is most frequently set to 2, referring to *binary* tournament selection.

Good examples to imagine the possible exploitation advantage of selective pressure are the functions that we considered in subsection 1.4.3, i. e., the Griewank function and modified the sinc function. Here, good mutants are likely to tend towards the global optimum (the zero vector) such that an increasing proportion of the good individuals will gather within its neighborhood during the evolution process. Then, selective pressure can accelerate the convergence to that global optimum.

1.5.4 Genotype Specific Variation Operators

Depending on the problem type there are different evolutionary representations and generation schemes for problem solutions. We deal with binary coded and real coded problems ($\mathcal{G} \in \{\{0,1\}^n, \mathbb{R}^n\}$) as well as with permutations ($\mathcal{G} = S_n$). Binary representation is natural if the actual problem depends on decision variables. Our two-coloring problem of Chapter 5 is of this kind. As mentioned above, binary coding is also possible (and has been extensively used) to cope with continuous problems. A disadvantage of this approach is that high precision requires long binary strings to represent each real number. According to Goldberg et al. [GDC92] (also see [Deb01]), the population size requirement will be also large in this case, which increases the computational effort. A second disadvantage is *hamming cliffs*, which are binary strings that represent neighbored values but differ in many bits (e. g., 011 and 100). For our real valued parameter optimization problem in Chapter 3, we therefore prefer an EA that operates on real numbers.

Concerning the TSP with time windows problem of Chapter 2, we operate on permutations which are mapped to solutions of the actual problem by an adapted insert heuristic. We introduce some of the established variation operators for the three kinds of genotypes.

Binary Variation Operators

- *Bit Flip Mutation.* Having a binary solution string, each bit is flipped with some small probability p_m to obtain a mutated solution.
- *Bit Swap Mutation.* If we want to retain the number of 0 (1) bits for some solution $x \in \{0,1\}^n$, we can uniformly choose an index i from the set $[n]_0 := \{k \in [n] \mid x_k = 0\}$ and an index j from the set $[n]_1 := \{k \in [n] \mid x_k = 1\}$ and swap the i-th bit with the j-th bit.
- *Single-Point Crossover.* Having two binary parent strings $x, y \in \{0,1\}^n$, a random locus i called *crossing site* is drawn from $[n]$ and all bits on the right site of the crossing site are exchanged between the parents yielding two offspring solutions.
- *Two-Point Crossover.* Here, instead of one crossing site, two crossing sites are chosen at random and the bits between both crossing sites are exchanged between the two parents. The process can be generalized to k-point crossover, choosing k crossing sites and exchanging alternate substrings between parents.
- *Uniform Crossover.* This operator chooses every bit of an offspring solution with probability 0.5 from either parent solution. This means an extreme case of k-point crossover.

Real Number Variation Operators

- *Uniform Mutation.* For a given $x \in \mathbb{R}^n$, uniform mutation chooses the mutation value of each component x_i under uniform distribution from an interval $[x_i - \delta_i, x_i + \delta_i]$, where the δ_i are suitable values (e. g., fractions of widths of possibly imposed box constraints).
- *Gaussian Mutation.* Here, a gaussian normal distribution $\mathcal{N}(x_i, \sigma_i^2)$ is taken as probability distribution for x_i. Suitable values have to be chosen for the variance σ_i^2 and may be adapted in each generation of the EA.

- *Blend Crossover (BLX-α).* This operator was suggested by Eshelman and Schaffer [ES92]. For two parents $x, y \in \mathbb{R}^n$, an offspring $z \in \mathbb{R}^n$ is generated by setting $z_i = \gamma_i x_i + (1 - \gamma_i) y_i$, $i \in [n]$, where the γ_i are randomly chosen from $[-\alpha, 1 + \alpha]$, $\alpha \geqslant -0.5$, under uniform distribution.

  In the extreme case of $\alpha = -0.5$, each z_i is simply the arithmetic average of x_i and y_i. For $\alpha = 0$, we deal with random convex combinations of the parent genes, while for $\alpha > 0$, an offspring gene z_i may also be outside the interval defined by x_i and y_i. Eshelman and Schaffer experimental found that $\alpha = 0.5$ is a good choice.

Permutation Variation Operators

Since a permutation $\pi \in \mathcal{S}^n$ has the unique vector representation $[\pi(i)]_{i=1}^n$, variation operators are described with respect to *permutation vectors*, i. e., vectors in $[n]^n$ with pairwise distinct components. We use some of the variation operators that are described in the recent book of Yu and Gen [YG10]. Clearly, all variation operators (must) maintain the properties of permutation vectors.

- *Exchange Mutation (EM).* Given a permutation vector $x \in [n]^n$, EM randomly chooses a locus i from $[n]$ and a locus j from $[n] \setminus \{i\}$ under uniform distribution. Then, the alleles at loci i and j are swapped.

- *Simple Inversion Mutation (SIM).* This operator reverses the order of some consecutive genes of a permutation vector. The consecutive genes are determined by the random choice of the minimum and maximum locus, respectively. If we represent solutions to the classical TSP problem by permutation vectors, SIM corresponds to a local search step of the 2-opt heuristic.

- *Partially Mapped Crossover (PMX).* For two parent permutation vectors $x, y \in [n]^n$ and random loci $i, j \in [n]$, $i \leqslant j$, the consecutive genes between locus i and locus j are exchanged between both parents to generate two temporary offsprings $u, v \in [n]^n$. Since u and v usually violate the permutation vector property, they are repaired as follows: For each gene outside the exchanged substring, it is checked weather its allele repeats within the exchanged substring. In this case, the allele is replaced by the corresponding allele of the former substring (that at the same locus w.r.t.

the substring). The procedure is repeated until the new allele of the gene does not occur within the exchanged substring.

- *Order Crossover (OX).* We use this operator in accordance to [Dav91], which is cited as its origin in [YG10] but has a somewhat different mode of operation, there. We are given two parent permutation vectors $x, y \in [n]^n$. The OX operator generates two offsprings $u, v \in [n]^n$ as follows. First, a random binary vector b of length n is generated, e. g., by randomized rounding of the vector with all values 1/2. Set $u_i = x_i$ for all $i \in [n]$ with $b_i = 1$. Then, for the complementary loci (where $b_i = 0$), the genes of u are assigned to the remaining alleles of x in the order they appear in y. Similarly, set $v_i = y_i$ for all $i \in [n]$ with $b_i = 0$ and assign the complementary genes of v with the remaining alleles of y in the order they appear in x. If OX operates on TSP solutions, it keeps the visiting orders of the inherited cities w.r.t. both parents.

1.5.5 Groups and Migration

The population P that is processed by an EA can be subdivided into *groups* to which selection and variation operators are applied locally. Interaction between groups is realized by *migration* operators that are periodically applied after a number of generations. A simple migration operation is to *clone* the very best individual over all groups into each group and to remove the worst individual of each group, subsequently. The concept of processing groups and doing migration from time to time is suitable for easy parallel EA implementations on distributed computer systems. We simply can use one CPU per group. To realize the migration principle, we will use the *Message Passing Interface (MPI)* [mpi] technology since it is commonly available.

1.5.6 Quantum Inspired Evolutionary Algorithms

Quantum inspired evolutionary algorithms (QiEAs) origin from the idea to use quantum computational principles for EAs. But rather than dealing with theoretical EA implementations in a quantum computer environment, QiEAs aim to benefit from related principles using conventional digital computers. A recent survey of QiEAs is given in [Zha11]. We will apply the QiEA framework proposed by Han and Kim [HK02][HK04] in Chapter 5.

Estimation of Distribution Algorithms

By now, the QiEA framework has been classed with *estimation of distribution algorithms (EDAs)* (see e. g., [DPSK09]). Regarding a classical EA with population size N_P in some i-th generation, each subset of the set of genotypes $\mathcal{G}$ with N_P elements will have a certain probability to be the population of generation $i+1$. The accordant probability distribution on $\binom{\mathcal{G}}{N_P}$ is implied by the randomness of the selection and variation operators. The idea behind EDAs is to learn an explicit probability distribution from selected individuals of the current population, which in turn is directly sampled to build the population of the succeeding generation.

Sampling Probability Distributions by Observing Quantum Bits

The QiEA framework of Han and Kim is intended to solve binary optimization problems and makes use of explicit probability distributions that correspond to a *quantum bit (qubit)* representation for the genes of individuals. In quantum computing, a qubit q is a tuple $q = (\alpha, \beta) \in \mathbb{C}^2$ with $|\alpha|^2 + |\beta|^2 = 1$ which degenerates (*collapses*) to a binary number $b \in \{0, 1\}$ in the moment of *observation*, namely to 0 with probability $|\alpha|^2$ respectively 1 with probability $|\beta|^2$. One solution $x \in \{0, 1\}^n$ of a binary optimization problem can be obtained from an n-dimensional qubit

$$Q = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_i & \cdots & \alpha_n \\ \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_i & \cdots & \beta_n \end{pmatrix},$$

by component-wise observation. Having a probability vector $p \in [0, 1]^n$ and choosing (α_i, β_i) such that $|\alpha_i|^2 = 1 - p_i$ and $|\beta_i|^2 = p_i$ for $i \in [n]$, this multi-dimensional observation means nothing else than sampling the solution x by randomized rounding, i. e., $x = \mathrm{RR}(p)$ (see Section 1.4.2).

Quantum Inspired Estimation of Distribution

A quantum gate acting on $\mathbb{C}^n$ is a unitary matrix $C \in \mathbb{C}^{n \times n}$. Thus, multiplication of a qubit $(\alpha, \beta) \in \mathbb{C}^2$ with a 2-dimensional quantum gate yields a new qubit providing new probabilities for the outcome of an observation. The QiEA framework of Han and Kim considers 2-dimensional rotation gates, which are 2×2-matrices $C_{\Delta\theta}$ representing rotation in $\mathbb{C}^2$, where

$$C_{\Delta\theta} = \begin{pmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{pmatrix},$$

and $\Delta\theta$ is the rotation angle.

Having two qubits $(\alpha, \beta), (\tilde{\alpha}, \tilde{\beta})$ with $|\beta|^2 = |\tilde{\beta}|^2$ and using the same rotation matrix to obtain $(\alpha', \beta') = C_{\Delta\theta}(\alpha, \beta)^T$ and $(\tilde{\alpha}', \tilde{\beta}') = C_{\Delta\theta}(\tilde{\alpha}, \tilde{\beta})^T$ it holds that $|\beta'|^2 = |\tilde{\beta}'|^2$. Thus, if we want to change a probability vector $p \in [0,1]^n$, we may consider for $i \in [n]$ the real values $\alpha(p_i) = \sqrt{1 - p_i}$ and $\beta(p_i) = \sqrt{p_i}$, apply the rotation matrices

$$\begin{pmatrix} \alpha'(p_i) \\ \beta'(p_i) \end{pmatrix} = C_{\Delta\theta_i} \begin{pmatrix} \alpha(p_i) \\ \beta(p_i) \end{pmatrix}$$

and obtain the new probability vector p' by setting $p'_i = \beta'(p_i)^2$. The pairs $(\alpha(p_i), \beta(p_i))$ represent points on the unit circle in the plane. The angle between $(\alpha(p_i), \beta(p_i))$ and the point $(1, 0)$ is given by $\theta(p_i) = \arctan(\frac{\beta(p_i)}{\alpha(p_i)})$ and lies in $[0, \pi/2]$. The corresponding angle of p'_i will be $\theta(p_i) + \Delta\theta_i$ (see left hand side of Figure 1.5).

Suppose we are given a probability vector $p \in [0,1]^n$ and a binary sample $x \in \{0,1\}^n$ with a good objective function value $f(x)$ (according to an optimization problem). Now, we aim to modify p such that it is more likely to yield x by randomized rounding. This can be done by changing the values of the components of p slightly towards the corresponding values of x. We also say that x *attracts* p. Rather than using an additive learning rate parameter Δ for changing the probabilities, QiEA applies the discussed rotation by an angle $\Delta\theta$ using suitable orientation. The rotation operation has less effect on a probability p_i if it is close to 0 or 1, as we have $p_i = \sin^2(\theta(p_i))$ (see right hand side of Figure 1.5). The procedure yields a particular estimation of distribution.

QiEA Terminology

Dealing with one probability vector $p \in [0,1]^n$ only, the observation and estimation (variation) process described above does already comprise both

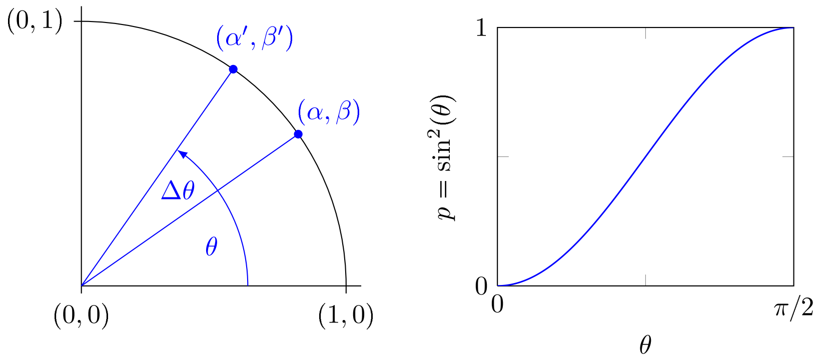


Figure 1.5. For a given probability $p \in [0,1]$ rotation of the unit circle point $(\alpha, \beta) = (\sqrt{1-p}, \sqrt{p})$ by an angle $\Delta\theta$ results in a new point (α', β') on the unit circle and corresponding probability $p' = (\beta')^2$. A change in angle has higher effect to the corresponding probability p, if p is close to 0.5 and less effect if p is close to 0 or 1, respectively.

properties of classical EA populations: (initial) exploration capacity and learning ability. However, the QiEA framework of Han and Kim allows two more evolutionary levels by allowing populations of groups of individuals $\mathcal{I} = (p, a)$, each consisting of a probability vector p and a sample a of p. The sample a is used to store good solutions and to modify the distribution p as described above. For that reason, a is called the *attractor* of individual $\mathcal{I} = (p, a)$. In each generation of the QiEA the samplings of the individuals are updated by observing the corresponding distributions and than the distributions are updated w.r.t. the attractor.

Han and Kim use migration of *elitist* attractors, i. e., either the very best attractor of the whole population (global migration) or the very best attractor of a group (local migration) will replace the attractor of each individual in the population or group, respectively. Note, that the corresponding distributions do not migrate together with the attractors, here. Empirical studies in [HK03] suggest local migration at every generation but infrequent global migration (e. g., every 100th to 150th generation if 1000 generations are performed in total).

1.5.7 Operational Notations

We will use the following notations for the operational parameters of our EA frameworks.

- N_P: population size
- N_{gr}: number of groups
- N_{ind}: number of individuals per group (all groups are supposed to have equal size $\Rightarrow N_P = N_{\mathrm{gr}} N_{\mathrm{ind}}$)
- N_{mate}: number of matings per group and iteration (classical EA)
- N_{obs}: number of observations per individual and iteration (QiEA)

1.6 Algorithm Engineering

In the former sections we introduced theoretical and practical aspects of algorithm design. After the foundation of complexity theory, research has considered both aspects rather separately. Theory focused on abstract problems and the question if polynomial time algorithms exist for them. Practitioners had to find sophisticated implementations and data structures in order to obtain practically fast algorithms for certain real world problems. Their work was supported by the insight of experimental studies. Theoretical and practical efficiency, however, are not always as close related as desirable. For example, the theoretically efficient ellipsoid algorithm for solving linear programs has a polynomial worst running time but is impracticable for real world instances while the simplex algorithm is practically fast for most problems but not polynomial for worst case scenarios. Another example is the classical traveling salesman problem, which is $\mathcal{NP}$-complete. Despite this fact, practical algorithms are by now able to exactly solve real world instances of the classical TSP with more than 10000 cities.

The *algorithm engineering* paradigm arose circa 15 years ago and aims to join theory and practice by the interplay of experimental impressions and theoretical conjectures. A recent monograph on algorithm engineering is the book of Müller-Hannemann and Schirra [MHS10]. Practice is supposed to help theory by giving more experimental evidence for theoretical estimates and even ideas concerning expedient analysis steps.

Taking the two-coloring problem of the hypergraph $\mathcal{H} = (V, \mathcal{E}) = ([n], \mathcal{A}_n)$ in Chapter 5 as an example, the optimum of the problems objective function has been shown to be $\mathcal{O}(\sqrt[4]{n})$ but the proof of this result does not yield a practical algorithm to calculate such two-colorings. The best known practical algorithm for the problem guarantees its solution objectives to be $\mathcal{O}(\sqrt[3]{n \log n})$. We will therefore utilize the features of this algorithm to find an improved optimization framework for the problem. A suitable running time of our non-deterministic algorithm incorporates the number $|\mathcal{E}|$ of hyperedges. We proofed that $|\mathcal{E}|$ is $\Theta(n^2 \log n)$ after we have checked the conjecture experimentally.

Chapter 2

TSP with Multiple Time Windows

2.1 Introduction

Tour planning plays an important role in logistic practice. In particular, it belongs to the central issues of companies providing field service. Facing increasing traffic system costs in the near future, the application of efficient tour planning will become an even more crucial factor.

2.1.1 Our Problem

The topic of multiple time windows occurs in the following specific problem arising in commercial practice. This problem has been posed by the leading company in Germany for complex service tour planning.

A salesman (which in practice is an employee of a company) has to schedule a trip that lasts several days (usually one week) and visits several customers with respect to their opening hours, which can be considered as multiple time windows. On the first day the salesman starts at location S, called the depot. We assume that the salesman's home is close to the depot and that the trip from the depot to home does not cause additional costs. Similarly, we assume that there is a hotel close to each customer's location. At the end of each day, the salesman has two options. First, he may return to the depot S, stay overnight at home and start the next working day again at the depot. Second, he stays overnight at a hotel, which for each night costs a fixed fee, and serves the next customer on the next day. Note that hotel stays are allowed only on specified days. Between two customers and between a customer and the depot, a wage is charged per time unit and traveling expenses per distance unit. Each customer requires to be served within specified time windows, and an individual penalty fee is assigned to each customer, if he remains unserved. The salesman's maximal regular working time for each day is fixed and lies within a specified time window. A maximal working time is given for each day, possibly extended by some overtime, where each overtime causes extra premium. Furthermore because of holidays, some days in the period may not be allowed to be a trip day. Note that all times are given in minutes, and the times of day are minutes starting from midnight before the first working day. We call this earliest possible starting time by EPST. Overall we have to schedule a trip with minimum total costs using the following instance parameters:

- $n \in \mathbb{N}$: number of customers.
- $c_{ij} \in \mathbb{R}_{\geqslant 0}$ for $i, j \in [n+1]$, $i \neq j$: distance units between customer/depot i and customer/depot j, where the depot is represented by the number $n+1$.
- $d_{ij} \in \mathbb{R}_{\geqslant 0}$ for $i, j \in [n+1]$, $i \neq j$: ride time between customer/depot i and customer/depot j, where the depot is represented by the number $n+1$.
- $p_i \in \mathbb{R}_{\geqslant 0}$ for $i \in [n]$: service time required by customer i.
- $\tau_i \in \mathbb{N}$ for $i \in [n]$: number of time windows provided by customer i.
- $[a_{il}, b_{il}]$ for $i \in [n]$; $l \in [\tau_i]$, where $a_{il}, b_{il} \in \mathbb{R}_{\geqslant 0}$ with $a_{il} \leqslant b_{il}$: time windows provided by customer i (measured from EPST).
- $\sigma_i \in \mathbb{R}_{\geqslant 0}$ for $i \in [n]$: penalty fees for omitted customer i.
- $w \in \mathbb{N}$: number of working days.
- $m_l \in \mathbb{R}_{\geqslant 0}$ for $l \in [w]$: maximal regular working time for day l.
- $U \in \mathbb{R}_{\geqslant 0}$: maximal overtime per working day.
- $[\alpha_l, \beta_l]$ for $l \in [w]$, where $\alpha_l, \beta_l \in \mathbb{R}_{\geqslant 0}$ with $\alpha_l \leqslant \beta_l$: daily working time windows (measured from EPST).
- $\gamma \in \mathbb{R}_{\geqslant 0}$: traveling expenses (per distance unit).
- $\delta \in \mathbb{R}_{\geqslant 0}$: wage (per time unit).
- $\mu \in \mathbb{R}_{\geqslant 0}$: overtime premium (per time unit).
- $\lambda_l \in \mathbb{R}_{\geqslant 0}$ for $l \in [w-1]$: hotel fee for day l.

  Note that $\lambda_l = +\infty$ means that a hotel stay is forbidden between the days l and $l+1$.

2.1.2 Related Work

Many problems from tour planning can often be modeled by variations of the famous Traveling Salesman Problem (*TSP*) [ABCC06][GP02], where for a given set of cities and distances between each pair of them, we have to find a shortest tour traversing each city exactly once. One such variation is the TSP with time windows (*TSPTW*) [AFG00][AFG01], where the distances between two vertices are given in terms of time and each vertex is assigned to a time interval in which the salesman has to arrive. Exact method contributions w.r.t. the TSPTW were given by Ascheuer, Fischetti and Grötschel who gave an ILP formulation and investigated the underlying polyhedra [AFG00]. They also solved problem instances by a proposed branch and cut algorithm [AFG01]. A further generalization is the TSP with multiple time windows (*TSPMTW*) [PGPR99]. There are also some papers about heuristics for the TSPTW like variable neighborhood search [dSU10] and insertion [GHLS98]. Contributions about evolutionary algorithms rather concern the classical TSP [Pot96][MMP05][ALED08]. A comparative study is carried out in [BB09].

2.1.3 Complexity of the TSPTW

The standard TSPTW is a subproblem of ours. It is $\mathcal{NP}$-hard as it contains the classical TSP as a subproblem. Savelsberg [Sav85] showed that even finding a feasible solution of the TSPTW is $\mathcal{NP}$-complete. The TSPTW also contains the TSP with deadlines problem (see 1.4.1) as a subproblem. This implies, that it does not allow for a polynomial algorithm with constant factor approximation (unless $\mathcal{P} = \mathcal{NP}$) in general.

2.2 A MILP for the Problem

The following MILP model is a variation of the model described in the Diploma Thesis of Mourad El Ouali [EO07]. We introduce some auxiliary variables which enable us to change some linear constraints and to discard some other linear constraints. The changes allow for an easier verification of the natural meaning of the MILP.

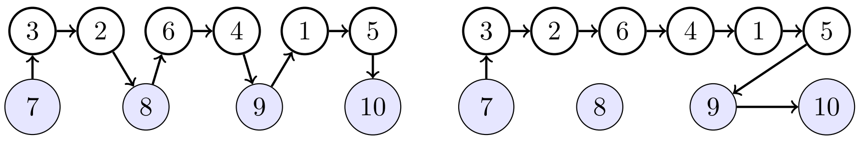


Figure 2.1. Two possible trips for an instance with $n = 6$ and $w = 3$.

We use a directed graph $G = (V, E)$ with $V = [n + w + 1]$ to represent the problem described in subsection 2.1.1. For $i = 1, 2, \ldots, n$, each customer i corresponds to a vertex i. The depot S is added as vertex $n + 1$. As the salesman may return to the depot after each day, we add an extra depot vertex $n + l + 1$ for each day $l \in [w]$. The arc set E consists of the following arcs:

- $(n + l, i)$ for $i \in [n]$; $l \in [w]$: The salesman starts at the depot at day l and then serves customer i.
- $(i, n + l + 1)$ for $i \in [n]$; $l \in [w]$: The salesman serves customer i at day l and then returns to the depot.
- (i, j) for $i, j \in [n]$, $i \neq j$: The salesman serves customer i, and after that he serves customer j.
- $(n + l, n + l + 1)$ for $l \in [w]$: The salesman stays at home at day l.

For this graph we seek for a path P from vertex $n+1$ to vertex $n+w+1$
.

2.1 Example. Figure 2.1 shows two example paths for an instance with $n = 6$, $w = 3$. The first path corresponds to a solution, where the salesman serves customers at each day and also returns to the depot at each day. In the second path, the salesman stays at a hotel close to customer 4 at the first day, then serves the remaining customers and returns to the depot at the second day. Finally, he stays at home at the third day.

In order to include all required factors (number of customers, distances, ride times, service times, hotel options, penalty fees, overtimes etc.) as optimization parameters in the MILP, we make use of the following variables:

$$x_{ij} := \begin{cases} 1, & \text{if } (i,j) \in P \\ 0, & \text{otherwise} \end{cases} \quad \text{for } (i,j) \in E$$

$$y_{il} := \begin{cases} 1, & \text{if the } l\text{-th time window of customer } i \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$
$$\text{for } i \in [n];\ l \in [\tau_i]$$

$$h_{il} := \begin{cases} 1, & \text{if the salesman stays overnight at a hotel close to customer } i \\ & \text{between day } l \text{ and day } l+1 \text{ after serving customer } i \\ 0, & \text{otherwise} \end{cases}$$
$$\text{for } i \in [n];\ l \in [w-1]$$

$$h'_{il} := \begin{cases} 1, & \text{if the salesman stays overnight at a hotel close to customer } i \\ & \text{between day } l \text{ and day } l+1 \text{ before serving customer } i \\ 0, & \text{otherwise} \end{cases}$$
$$\text{for } i \in [n];\ l \in [w-1]$$

$$s_i := \begin{cases} 1, & \text{if customer } i \text{ is omitted} \\ 0, & \text{otherwise} \end{cases}$$
$$\text{for } i \in [n]$$

t_i := arrival time at customer $i \in [n]$ (measured from EPST)
t'_i := starting time of service at customer $i \in [n]$ (measured from EPST)
t''_i := leaving time from customer $i \in [n]$ (measured from EPST)
q_l := starting time of work at day $l \in [w]$ (measured from EPST)
e_l := end time of work at day $l \in [w]$ (measured from EPST)
u_l := overtime at day $l \in [w]$

Note that some values may become irrelevant in some cases, e. g., if the customer i for $i \in [n]$ is omitted, then the arrival time at customer i is not considered. In such cases, these values may be arbitrary.

Now, the minimization function of the overall costs is

$$\min \quad \sum_{i=1}^{n} \sigma_i s_i + \gamma \cdot \Big(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} \sum_{l=1}^{w} (c_{n+1,i} x_{n+l,i} + c_{i,n+1} x_{i,n+l+1}) \Big)$$

$$+ \sum_{i=1}^{n} \sum_{l=1}^{w-1} \lambda_l (h_{il} + h'_{il}) + \sum_{l=1}^{w} (\delta(e_l - q_l) + \mu u_l). \tag{2.1}$$

In order to include all the specifications mentioned above, we have to add several linear constraints to the objective function. We first force the starting vertex to have exactly one successor and the destination vertex to have exactly one predecessor in the path P:

$$x_{n+1,n+2} + \sum_{i=1}^{n} x_{n+1,i} = 1, \tag{2.2}$$

$$x_{n+w,n+w+1} + \sum_{i=1}^{n} x_{i,n+w+1} = 1. \tag{2.3}$$

Furthermore, each further vertex from $V \backslash \{n+1, n+w+1\}$ must have as much successors as predecessors in the path P:

$$\sum_{(i,j) \in E} x_{ij} = \sum_{(j,i) \in E} x_{ji} \quad \text{for } i \in V \backslash \{n+1, n+w+1\}. \tag{2.4}$$

The omission of a customer means that the corresponding vertex has no arrival arc:

$$s_i + \sum_{(j,i) \in E} x_{ji} = 1 \quad \text{for } i \in [n]. \tag{2.5}$$

If a customer is visited, this must happen within exactly one active time window:

$$s_i + \sum_{l \in [\tau_i]} y_{il} = 1 \quad \text{for } i \in [n]. \tag{2.6}$$

The salesman does not stay overnight at the hotel close to an omitted customer.

$$s_i + \sum_{l=1}^{w-1} h_{il} \leqslant 1 \quad \text{for } i \in [n]. \tag{2.7}$$

$$s_i + \sum_{l=1}^{w-1} h'_{il} \leqslant 1 \quad \text{for } i \in [n]. \tag{2.8}$$

At the end of each working day, the salesman stays overnight either at a hotel or at home:

$$x_{n+l,n+l+1} + \sum_{i=1}^{n} (x_{i,n+l+1} + h_{il} + h'_{il}) = 1 \quad \text{for } l \in [w-1]. \tag{2.9}$$

For each day l, the starting time of work is not larger than the end time of work.

$$q_l \leqslant e_l \quad \text{for } l \in [w]. \tag{2.10}$$

The actual working time at a trip day l starts at a time q_l and ends at a time e_l within the corresponding working time window:

$$q_l \geqslant \alpha_l \quad \text{for } l \in [w], \tag{2.11}$$

$$e_l \leqslant \beta_l \quad \text{for } l \in [w]. \tag{2.12}$$

The actual working time of each day l does not exceed the corresponding maximal value m_l by more than the overtime u_l:

$$e_l - q_l \quad \leqslant \quad m_l + u_l \quad \text{for } l \in [w]. \tag{2.13}$$

All overtimes are limited by U:

$$u_l \quad \leqslant \quad U \quad \text{for } l \in [w]. \tag{2.14}$$

The following conditions ensure that all arrival, leaving, starting and end times are in chronological order for all cases. Note that (2.17) – (2.24) are based on the fact that β_w is the end time of the last working day and that β_w is multiplied by -1 or 0. If the traveler serves a customer, the arrival time is not larger than the corresponding starting time of service:

$$t_i \quad \leqslant \quad t'_i \quad \text{for } i \in [n]. \tag{2.15}$$

If the traveler serves a customer, the end time of service is not larger than the corresponding leaving time:

$$t'_i + p_i \quad \leqslant \quad t''_i \quad \text{for } i \in [n]. \tag{2.16}$$

For each customer i, the arrival time at the next customer j in the path is not smaller than the leaving time at i plus riding time:

$$t_j \geqslant t''_i + d_{i,j} \cdot x_{ij} + (x_{ij} - 1) \cdot \beta_w \quad \text{for } i \neq j \in [n] \tag{2.17}$$

For each day l starting from the depot, the arrival time at the next customer i is not smaller than the begin of work plus riding time:

$$t_i \geqslant q_l + d_{n+1,i} \cdot x_{n+l,i} + (x_{n+l,i} - 1) \cdot \beta_w \\ \text{for } i \in [n];\ l \in [w-1]. \tag{2.18}$$

For each day l returning to the depot, the end time of work is not smaller than the leaving time at the last customer i plus riding time:

$$e_l \geqslant t''_i + d_{i,n+1} \cdot x_{i,n+l+1} + (x_{i,n+l+1} - 1) \cdot \beta_w \\ \text{for } i \in [n];\ l \in [w]. \tag{2.19}$$

For each day $l+1$, if the first customer i of the day is served after staying overnight at the hotel close to customer i, the starting time of service at i is not smaller than the starting time of work:

$$t'_i \geqslant q_{l+1} + (h'_{il} - 1) \cdot \beta_w \quad \text{for } i \in [n];\ l \in [w-1]. \tag{2.20}$$

For each day $l+1$, if the last customer i of the previous day has been served before staying at the hotel close to customer i, the leaving time at i is not smaller than the starting time of work:

$$t''_i \geqslant q_{l+1} + (h_{il} - 1) \cdot \beta_w \quad \text{for } i \in [n];\ l \in [w-1]. \tag{2.21}$$

For each day l, if the last customer i of the day is served before staying at the hotel close to customer i, the end time of work is not smaller than the end time of service at i:

$$e_l \geqslant t'_i + p_i + (h_{i,l} - 1) \cdot \beta_w \\ \text{for } i \in [n];\ l \in [w-1]. \tag{2.22}$$

For each day l, if the first customer i of the next day is served after staying at the hotel close to customer i, the end time of work is not smaller than the arrival time at i:

$$e_l \geqslant t_i + (h'_{i,l} - 1) \cdot \beta_w \qquad \text{for } i \in [n];\ l \in [w-1]. \tag{2.23}$$

The following conditions guarantee, that all arrivals occur within the chosen time windows of the visited customers:

$$t'_i \geqslant \sum_{l=1}^{\tau_i} a_{il} y_{il} - s_i \beta_w \quad \text{for } i \in [n]. \tag{2.24}$$

$$t'_i + p_i \leqslant \sum_{l=1}^{\tau_i} b_{il} y_{il} + s_i \beta_w \quad \text{for } i \in [n], \tag{2.25}$$

Note that all these constraints are necessary for the correctness of the MILP model.

2.3 A Problem Specific Branch and Bound Algorithm

A general BnC as used in many MILP solvers like CPLEX [cpl] might not be effective for such a special MILP, as introduced in Section 2.2, as it does not make use of its special structure. Therefore, in this section we suggest a problem based BnB for our MILP.

Compare a sub-path of a feasible path which has the form

$$X \quad := \quad (n+1, p_1, p_2, \ldots, p_t, q)$$

with $2 \leqslant t \leqslant n + w - 1$, $q \in \{1, 2, \ldots, n, n+2, n+3, \ldots, n+w+1\}$, $p_i \in \{1, 2, \ldots, n, \text{n+2}, \text{n+3}, \ldots, n+w\}$ for $i = 1, 2, \ldots, t$. Consider a fixed feasible permutation π on the set $\{1, 2, \ldots, t\}$ and the corresponding path

$$Y \quad := \quad (n+1, p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(t)}, q),$$

where π is feasible, if for all r, s with $1 \leqslant r < s \leqslant t$ with $n+1 \leqslant p_{\pi(r)}, p_{\pi(s)} \leqslant n + w + 1$ it holds: $p_{\pi(r)} < p_{\pi(s)}$. Clearly, both sub-paths lead to sub-trips of the salesman. If a complete path Z (ending with $n + w + 1$) starts with a given sub-path X, then Z is called *extended complete path* of X. The question is whether a sub-path has an extended complete path representing an optimal trip. In the following we will introduce a comparison criterion between two paths, X and Y.

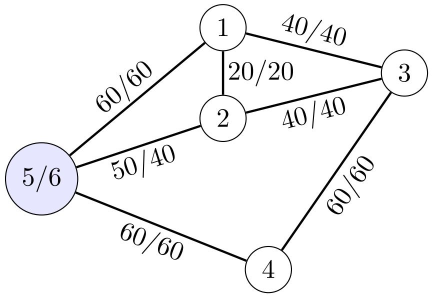
We observe that a sub-path does not uniquely determine the overall costs of the sub-trip, as the choice which time windows are active and the starting times q_l for each day l with $l \in [w]$ may affect the costs of the trip, too. For a sub-trip g, the starting times of each day and the departure times of the vertices are chosen such that the salesman reaches all possible time windows and waiting times are minimized. Let $c(g)$ be the accordant costs, which are minimal for g. There is a minimum departure time $t(g)$ and a (may be identical) maximum departure time $t'(g)$ for the (common) last vertex of the sub-path, such that costs remain $c(g)$. The time $t'(g)$ is obtained from $t(g)$ by adding the maximum possible service delay within the active time windows of the customers of the current trip day. We proceed as follows. First, we collect all possible trips resulting from the sub-paths X and Y, and receive the corresponding sets C_X and C_Y. Second, we test for each sub-trip h of C_Y, whether a sub-trip g of C_X exists with the following two conditions:

(a) $c(g) < c(h)$,

(b) $t(g) \leqslant t(h)$.

Both steps do not cost much time, as usually in practice the number of possible time windows is small for each customer. If the second criterion is fulfilled, then it seems reasonable that all extended complete paths of Y cannot represent an optimal trip and thus can be excluded from the further search. Surprisingly, this is not the case, which can be observed in Example 2.2.

2.2 Example. Suppose we have to serve four customers within one day, i. e., $n = 4$ and $w = 1$. Each customer requires a service of 30 minutes. The daily opening hours of the customers as well as the distances in kilometer (km) and riding times in minutes are shown in Figure 2.2, where edge labels have the form km/minute. The traveling expenses per km are set to 1 cost unit as well as the wage per minute. The distance of the sub-path $X = 5, 1, 2, 3$ is 120 km, and its minimum cost is $c(X) = 330 = 120 + 120 + 90$. The sub-trip described by the sub-path $Y = 5, 2, 1, 3$ is 110. i. e., 10 km shorter, and the minimum cost of Y is $c(Y) = 300 = 100 + 110 + 90$. A further parameter is the latest possible finish time, such that all time windows can



Customer	Opening hours
1	8:00-12:00
2	8:00-12:45
3	10:00-14:00
4	15:00-18:00

Figure 2.2. Distance map and time windows for an instance with $n = 4$ and $w = 1$.

still be met. In the case of X, the time window of customer 2 enforces a tour start not later than 10:25 h. This implies that the latest possible finish time at customer 3 will be 13:55 h. In the case of Y we obtain 10:00 h as latest possible starting time and 13:10 h as latest possible finish time at customer 3. This time is enforced by the time window of customer 1. Thus Y beats X w.r.t. three parameters: finish time, distance and minimum costs. Concerning the unique completions $\overline{X} = 5, 1, 2, 3, 4, 6$ and $\overline{Y} = 5, 2, 1, 3, 4, 6$, the afternoon time window of customer 4 causes waiting times. We obtain the settings shown in Table 2.1. The remaining waiting time for $\overline{Y}$ is 45 minutes longer than that for $\overline{X}$. The required working time is 365 minutes for $\overline{X}$ and 390 minutes for $\overline{Y}$. The distances are 240 km and 230 km for $\overline{X}$ and $\overline{Y}$, respectively. Thus, we have $c(\overline{X}) = 605 < 620 = c(\overline{Y})$, and $\overline{X}$ is the optimal path.

The reason for this behavior is that possible waiting times are not considered in the above criterion. We may remedy this as follows. If the common last vertex q of both paths is a customer vertex, we denote by $d(g)$ $(d(h))$ the day of the chosen time window for q in g (h). If q is an extra depot vertex, we set $d(g) = d(h) = q - n - 1$, i. e., we deal with the corresponding working day "before" q. We further denote by $w(g)$ $(w(h))$ the total working time of the traveler at day $d(g)$ $(d(h))$. Now, we change

Table 2.1. Optimum schedules of two feasible trips for an instance with $n = 4$ and $w = 1$.

$\overline{X}$				$\overline{Y}$			
No.	Arr.	Service	Dep.	No.	Arr.	Service	Dep.
5			10:25	5			10:00
1	11:25	11:25-11:55	11:55	2	10:40	10:40-11:10	11:10
2	12:15	12:15-12:45	12:45	1	11:30	11:30-12:00	12:00
3	13:25	13:25-13:55	13:55	3	12:40	12:40-13:10	13:10
4	14:55	15:00-15:30	15:30	4	14:10	15:00-15:30	15:30
6	16:30			6	16:30		

criterion (a) to (a'):

$$\begin{array}{ll} c(g) + \max(0, t'(h) - t'(g)) \cdot (\delta + \mu) < c(h), & \text{if } q \in [n] \text{ and } d(g) = d(h) \\ c(g) + (d(h) - d(g)) \cdot \lambda < c(h), & \text{if } q \in [n] \text{ and } d(g) \neq d(h) \\ \text{true}, & \text{if } q \geqslant n + 2, \end{array}$$

and introduce criterion (c):

$$\begin{array}{ll} w(g) \leqslant w(h), & \text{if } q \in [n] \text{ and } d(g) = d(h) \\ \text{true}, & \text{else} \end{array}$$

It holds the following

2.3 Lemma. *Suppose, that for every subtrip $h \in C_Y$ there is a subtrip $g \in C_X$ such that (a'), (b) and (c) hold for g and h. Then, no extended complete path of Y corresponds to an optimal trip.*

Proof. Assume there is an extended complete path $\overline{Y} = (Y, r_1, r_2, \cdots, r_s)$ of Y that has an associated optimal trip $\overline{h}$. Consider the restriction $h \in C_Y$ of $\overline{h}$ to Y and choose $g \in C_X$ such that (a'), (b) and (c) hold for g and h. Construct the completion $\overline{X} = (X, r_1, r_2, \cdots, r_s)$ of X and choose $\overline{g} \in C_{\overline{X}}$ such that the customer vertices in $\{r_1, \cdots, r_s\}$ have the same time windows and service times as in $\overline{h}$. From (b) and (c), it follows that $\overline{g}$ is a valid trip. We further see that, if at least one of the vertices q and r_1 is an extra depot vertex, the cost advantage of g in comparison to h is equal to the cost advantage of $\overline{g}$ in comparison to $\overline{h}$. Now, we consider the case that both vertices q and r_1 are customer vertices. If $d(g) = d(h)$, the cost of $\overline{g}$ in comparison to $\overline{h}$ may only be increased by a maximum waiting time of $\max(0, t'(h) - t'(g))$ before r_1 multiplied with the maximum costs for this working time, which in case of overtime is $\delta + \mu$. If $d(g) \neq d(h)$, maximum additional cost may be caused by fees λ for $d(h) - d(g)$ nights in hotel. Thus, criterion (a') implies that $\overline{g}$ is cheaper than $\overline{h}$, which contradicts our assumption. □

Our problem based BnB traverses all possible sub-paths which are the vertices of the search tree. Each sub-path begins with the initial depot vertex $n + 1$. A next unvisited vertex is added step by step. We use two pruning (backtracking) rules. The first one is the usual LP based rule, i. e., we solve the current LP relaxation. If the received lower bound is not smaller than the costs of the current best trip, this sub-path and its extended complete paths may be excluded. For the branching process, we prefer customer vertices to depot vertices, as omitting customers should occur only rarely and abidance in the depot should be avoided. We also prefer arcs whose value in the current LP solution is close to 1.

The second backtracking rule is our criterion concerning sub-paths. For this purpose we use a memory data structure that stores the relevant comparison information about the trips of already examined sub-paths that are not dominated by other trips of other examined sub-paths. The relevant comparison information about a trip $g \in C_P$ of a sub-path P is given by the end vertex of P, the set of vertices that are contained in P and the parameters $c(g)$, $t(g)$ and $t'(g)$. Since only trips of sub-paths are comparable that have both the end vertex and the set of traversed vertices in common, we subdivide our data structure into equivalence classes with respect to this property. For every suitable sub-path length (between 4 and $n + w$) we will store relevant trip information of already examined sub-paths for a maximum storage number of, say 100, equivalence classes. A currently examined path P does either have an equivalence class $[P]$ for which a memory entry does already exist or adds $[P]$ to our data structure (maybe replacing the oldest class for sub-paths of the same length, if the corresponding maximum number is reached). In the first case, the current cost/time information about all undominated trips for $[P]$ can be updated by the associated information about P. If for every trip $g \in C_P$ there is already information about a trip h that is superior to g, P will not change the information associated with $[P]$ and will thus be excluded from search. In our implementation we use the C++ standard containers `vector` and `list` and represent the sub-path classes as a vector of vectors, where each element is composed of the information about the class (vertex set and end vertex) and the list of (currently) undominated trips.

In order to give a pseudo code description of our BnB algorithm, we will use a structure T with the following fields

- LP: Linear program that represents the current relaxation of the MILP
- ub: Upper bound on the objective value of the MILP
- P: List of integers in $[n + w + 1]$ that represents the search path
- s_P: List with schedule information about all relevant trips that correspond to P (more precisely, each element of s_P contains the necessary schedule information about all trips that belong to the associated sub-path of P and might have an extension to an optimal trip)
- branchArcs: List with the LP indices of the arcs that have been used to reach the current branch of the search tree
- branchValues: List with the values from $\{0, 1\}$ that are currently chosen for the LP variables that correspond to branchArcs
- isBranchArc $\in \{\text{false}, \text{true}\}^{|E|}$: Indicator vector for the arcs that are currently in branchArcs
- inPath $\in \{\text{false}, \text{true}\}^{|V|}$: Indicator vector for the vertices in P
- P_{best}: List of integers that represent the best examined complete path
- s_{best}: schedule information of an optimal trip for P_{best}

and a structure PM for the sub-path memory as described above. The main algorithm uses the auxiliary routines initTree, updateTree, trackbackTree and updatePathMemory. As the names imply, initTree initializes the binary search tree with the root node while updateTree and trackbackTree perform forward steps and backward steps, respectively. A forward step to the next search tree node is done by adding a new vertex to the search path and selecting the corresponding arc, i. e., fixing its LP variable to 1. In a backward step, an arc that is currently selected will be forbidden by changing the fixing of its LP variable from 1 to 0 and removing the corresponding vertex from the search path. If the arc of the current search tree node is already forbidden, trackbackTree goes back to the first node of a selected arc and changes that fixing. On the way back, the search path gets shortened and the LP variables of former forbidden arcs have to be relaxed to the range $[0, 1]$, again. In the main procedure, line 12 integrates our second backtracking criterion. If we ignore this step, we deal with the general branch and bound method for integer programs described by Dakin [Dak65] but concerning the binary case. Our second backtracking criterion requires the scheduling information of relevant trips of both the search path (stored in $T.s_P$) and former examined sub-paths (stored in PM). The updating process of $T.s_P$ performed in line 22 of updateTree requires some coding effort as some case distinctions w.r.t. the properties of the current end vertex of $T.P$ and its predecessor have to be considered. Moreover, visit times that correspond to the time windows at the last day of a trip must be scheduled such that costs are minimized, i. e., possible waiting times are minimized by delaying the start of that working day, if applicable. However, updating T.sP is computational efficient in practice as the number of relevant trips remains moderate for each sub-path. The same holds for the stored trips of already examined paths in PM.

```
Algorithm 1: initTree
   input  : Instance parameters (see subsection 2.1.1) and upper bound ub
   output : Initial search tree
1  obtain $T$.LP by relaxing each boolean MILP variable to $[0, 1]$;
2  $T$.ub = ub;
3  initialize $T.P_{\text{best}}$ to represent the empty tour $(n + 1, n + w + 1)$;
4  initialize $T.P$, $T.s_P$, $T$.branchArcs and $T$.branchValues to be empty;
5  initialize all components of $T$.inPath and $T$.isBranchArc with false;
6  append the start vertex $n + 1$ to $T.P$;
7  append the list with the schedule information of the unique relevant
   trip $g \in C_P$ $(c(g) = 0,\ t(g) = \alpha_1,\ t'(g) = \beta_1)$ for $T.P$ to $T.s_P$;
8  $T$.inPath$(n + 1)$ = true;
9  append 0 to both lists branchArcs and branchValues;
10 return $T$;
```

Algorithm 2: updateTree

```
input  : search Tree $T$ (by reference)
output : boolean value that indicates if $T$ could be updated
1  let $u$ be the current end vertex of the search path $T.P$;
2  found = false;
3  $\omega_{\text{best}} = -1$;
4  $V_1 = \{v \in [n] \mid$ not $T.\text{inPath}(v)\}$;
5  $V_2 = \{v \in [n+w+1] \backslash [n] \mid$ not $T.\text{inPath}(v)\}$;
6  $i = 1$;
7  while $i \leqslant 2$ and not found do
8      $i = i + 1$;
9      for $v \in V_i$ do
10         $e = (u, v)$;
11         if not $T.\text{isBranchArc}(e)$ and $v$ can be included into $T.P$ such
           that at least one of the trips in $T.s_P$ can be extended to a new
           feasible trip then
12             found = true;
13             let $\omega$ be the value of the variable for the arc $e$ in the
               current solution of $T.\text{LP}$;
14             if $\omega > \omega_{\text{best}}$ then
15                 $v_{\text{best}} = v$;
16                 $\omega_{\text{best}} = \omega$;
17 if found then
18     $v = v_{\text{best}}$;
19     $e = (u, v)$;
20     append $v$ to $T.P$;
21     $T.\text{inPath}(v) = \text{true}$;
22     append a list of relevant trip schedules for the updated path $T.P$
       to $T.s_P$;
23     append $e$ to $T.\text{branchArcs}$;
24     $T.\text{isBranchArc}(e) = \text{true}$;
25     append 1 to $T.\text{branchValues}$;
26     fix value of the variable for $e$ in $T.\text{LP}$ to 1;
27 return found;
```

Algorithm 3: trackbackTree

```
input/output: search Tree $T$ (by reference)
1 while $T$.branchValues is not empty and last value of $T$.branchValues is 0 do
2     let $e$ be the last arc of $T$.branchArcs;
3     relax the variable for $e$ in $T$.LP to the range $[0, 1]$ again;
4     delete last arc from $T$.branchArcs;
5     delete last value from $T$.branchValues;
6     $T$.isBranchArc($e$) = false;
7 if $T$.branchValues is not empty then
8     let $e$ be the last arc in $T$.branchArcs;
9     replace last value 1 of $T$.branchValues by 0;
10    fix value of the variable for $e$ in $T$.LP to 0;
11    let $v$ be the last vertex in $T.P$;
12    $T$.inPath($v$) = false;
13    delete last vertex $v$ from $T.P$;
14    delete last element from $T.s_P$;
```

Algorithm 4: updatePathMemory

```
input/output: search tree $T$ and path memory PM (both by reference)
output      : boolean value that indicates if updating changed PM
1 changed = true;
2 if a sub-path class entry for $[T.P]$ exists in PM then
3     update the list of undominated trips for the class $[T.P]$ by
      comparison with the undominated trips for $T.P$ that are given by
      the last element of $T.s_P$;
4     if $[T.P]$ entry remains unchanged then changed = false
5 else
6     generate a new entry for the class $[T.P]$ that consists of the
      relevant information of the undominated trips for $T.P$;
7     Either add the new entry to PM or replace the "oldest" entry that
      concerns sub-paths of the same length as $T.P$;
8 return changed;
```

Algorithm 5: BnB algorithm for TSPTW

```
   input  : Instance parameters as described in subsection 2.1.1
   output : An optimal trip
1  $T$ = initTree;
2  while $|T.P| > 0$ and $t \leqslant t_{\max}$ do
3      update lower bounds on the time variables of $T$.LP w.r.t. the
       earliest finish time of $T.s_P$;
4      try to solve current $T$.LP;
5      if $T$.LP infeasible then trackbackTree($T$);
6      else
7          Let $c^*$ be the objective value of the current LP solution $x^*$;
8          if $c^* >$ ub then trackbackTree($T$);
9          else
10             if not updateTree($I, T$) then trackbackTree($T$);
11             else
12                 if not updatePathMemory($I, T$) then trackbackTree($T$);

13                 else if $P$ is complete then
14                     Let $c(g)$ be the cost of the unique trip $g$ for $T.P$
                       given by $T.s_P$;
15                     Obtain tour cost $c$ from $c(g)$ by adding the penalty
                       fees of unvisited customers;
16                     if $c < T$.ub then
17                         $T$.ub = $c$;
18                         $T.P_{\text{best}} = T.P$;
19                         obtain schedule $T.s_{\text{best}}$ of one optimal trip from
                           $T.s_P$;
20                     trackbackTree($T$);
21 return $T.P_{\text{best}} = T.P$ and $T.s_{\text{best}}$;
```

Since our algorithm extends the classical algorithm of Dakin [Dak65] by the backtracking criterion concerning sub-paths, we obtain from Lemma 2.3 that

2.4 Theorem. *The problem based branch und bound algorithm finds an optimal* *trip.*

2.5 Remark. In the moment when the search-path $T.P$ becomes a complete path from $n+1$ to $n+w+1$, we have already calculated the cost of the associated optimal trip. We do not have to branch on the indicator variables for the choice of time windows nor hotel options, since this task is implicitly concluded within the updating process of the search paths trips in $T.s_P$.

2.4 A Randomized Insert Heuristic

We chose a heuristic generalizing elementary techniques for the TSP as *Cheapest Insert* [RSL77] and *2-OPT* [LK73], and including a randomized component by using the restarting technique [GSK98]. It was introduced in [EO07]. We outline the main steps of the heuristic and then present some specific refinements and optimizations of the heuristic.

1. Consider the path $P := (n+1, n+w+1)$.
2. Generate a candidate list of customers in random order.
3. For each day l with $l = 1, 2, \ldots, w$ choose at random one of four options:

   (a) A trip is forbidden at day l.
   $\Rightarrow$ The vertices $n+l$, $n+l+1$ and the arc $(n+l, n+l+1)$ are included in the path.

   (b) After the work the traveler returns to the depot.
   $\Rightarrow$ The vertex $n+l+1$ is included in the path.

   (c) After the work he stays overnight at the hotel close to the last customer of this day.
   $\Rightarrow$ If $l \neq w$, the vertex $n+l+1$ is not present in the path.

   (d) After the work he stays overnight at the hotel close to the first customer of the next day.
   $\Rightarrow$ If $l \neq w$, the vertex $n+l+1$ is not present in the path.

   Note that the cases b), c), d) may lead to case a), if at the end of the procedure it is not possible to include a customer i between $n+l$ and $n+l+1$.

4. Obtain a starting path P.
5. For each day l with $l = 1, 2, \ldots, w$ choose at random the overtime u_l from the set $[0, U]$.
6. Start with the first customer vertex in the candidate list.
7. Try to insert this customer vertex at a certain place. Choose the place of insertion so that the costs are minimum over all places. Do this insertion only if the costs of the path become smaller by this insertion (recall that a penalty fee is assigned for omitting a customer).

   Note that if a customer has more than one possible time window at a day, then the earliest possible time window is chosen such that the tour remains valid with respect to the settings of step 3 resulting in a unique trip for the current path.
8. Update the candidate list be removing the current candidate.
9. GOTO 6.
10. Assume that in the current path at some day more than one customer is served, say the customers $C_1, C_2, \ldots C_t$ with $t \geqslant 2$. Then for $1 \leqslant i < j \leqslant t$, the order of the customers at this day is changed to $C_1, C_2, \ldots, C_{j-1}, C_i,\ C_{j+1}, \ldots, C_{i-1}, C_j,\ C_{i+1}, \ldots, C_{t-1}, C_t$, if all corresponding time windows can also be met by the new order and if the overall costs are decreased by this change. In other words, the positions of customers i and j are exchanged, if possible. Note that such a step can be viewed as an OPT step, where arcs contained in the path are replaced by further arcs not contained in the path.
11. Apply step 10 for all possible days with more than one customer and as long as such steps decrease the overall costs.
12. Repeat all steps 1 to 11 in multiple trials. In each trial a different random order of the customers and a different combination of options for each day is chosen.
13. Choose the best path of all considered trials.

Similar to the scheduling procedure in our branch and bound algorithm, the visit times that correspond to the current path and time windows are scheduled such that costs are minimized, i. e., possible waiting times are minimized by delaying working day starts, if applicable.

Further assume a customer has only few and small opening hours, and the penalty fee for omitting him is large. The situation might occur that the insertion step of the heuristic is tried relatively late such that the customer's opening hours cannot be reached by the salesman at this state. To avoid this situation we suggest to introduce a probability distribution, where such a customer is placed at the beginning of the customer's list with large probability.

If this situation still occurs and a customer i with large penalty fee cannot be included any more, we suggest further modifications. First, replace an already included customer j by the customer i. After that replacement it might be possible to include also customer j. Second, consider also OPT steps that slightly increase the costs of the trip, and test whether inserting the customer i after that change may be possible.

2.5 EA Framework

Being rather satisfied with the results of the random insert heuristic, we integrate its scheduling procedure (steps 6 to 11) as encoding function into our EA framework. Thus, our genotypes are composed of a permutation of $[n]$ that represents the scheduling candidate list (cf., step 2), a real valued string of length w that represents the maximum allowed overtimes u_l per day (cf., step 5), and a binary string of length $3w$ that represents the daily tour options (cf., step 3) where each option is represented by 3 bits indicating if

- a day is a working day or not,
- the traveler takes a hotel or returns to depot,
- a customer close to a chosen hotel is served in the evening or at the next morning.

The initial population consists of randomly generated individuals. We use linear ranking based roulette wheel selection with selective pressure 2 (see subsection 1.5.3) to find the mating candidates for crossover operations. We deal with every possible combination of the permutation variation operators EM, SIM, PMX and OX. For the binary parts we use bit-flip mutation of one randomly chosen gene (bit-swap mutation does not seam reasonable, here) and two-point crossover. Uniform mutation and BLX-α crossover are done for the real valued genes. Here, with probability 0.5 we mutate within a 10% tolerance range of the current values and use $\alpha = 0.25$ for crossover, retaining the gene limits in both cases. Choosing the best found combination of permutation variation operators, we further perform experiments with multiple groups.

2.6 Experiments

We have implemented all our algorithms described above in C++ and used the state-of-the-art package Cplex, version 12, to build and solve our MILP (Section 2.2) and its LP relaxations, respectively. In order to solve the MILP model we used both the Cplex MIP solver solitary and the corresponding Cplex dual simplex solver within our branch and bound algorithm of Section 2.3. We further have implemented the randomized heuristic of Section 2.4 and combined it as solution generation scheme with the EA framework of Section 2.5. Regarding the number r of trials in the heuristic, we observed in pre-experiments that $r = 100 \cdot n$ is a good choice. Thus we used this choice in the following experiments. All our experiments were carried out on a Linux system with AMD Opteron CPUs having 2100 MHz clock rate, 512 kB cache and 32 GB of RAM. Overall we investigated 99 test instances, namely 28 real-world instances of FLS [FLS], 8 self-created instances and 63 random instances. The real-world instances are denoted by the corresponding regions in Germany, where the customers of these instances are located, and by an additional number if we have several instances in this region. The first five self-created instances represent a fictive service for restaurants in Schleswig-Holstein, where each day contains three time windows. As further self-created test instances we considered a possible politician journey from Kiel to the 15 other regional German capitals. Berlin is visited a second time as federal capital of Germany, and Kiel represents the depot vertex. Additionally, we tested one sub-instance of this instance. The last self-created instance is a two days tour around Kiel. The random instances differ in three parameters, namely the number of customers n with values 5, 10, 15, 20, 30, 40, 50, the number of days w with values 3, 5, 7, and the length of a grid square, where all locations are uniformly and randomly chosen from, with values 100, 300, 500 kilometers.

The remaining parameters were chosen in a random, but realistic way.

All 99 test instances with the most important parameters number of customers ("Cu.") and number of trip days ("Da.") are listed in the Tables 2.2, 2.3 and 2.4. These tables also present the times of the Cplex MILP solver, the BnB algorithm and the heuristic, where all times are given in seconds and a time limit of 1 hour (3.600 seconds) was set for an experiment on a single instance. Fortunately, if the MILP and the BnB, respectively, was stopped after 1 hour, nevertheless upper bounds could be obtained, which are also listed in the tables. Additionally, also the values of the corresponding LP relaxations are listed as lower bounds ("LB"). Note that if such a lower bound is omitted in the table, this means that one of both exact methods terminated for this instance in less than 1 hour and verified the optimum value. Additionally, all verified optimum values appear in bold face in the tables.

Furthermore the upper bounds found by both exact methods and the heuristic are compared.

The results show that the MILP solver of Cplex was not able to exactly solve the MILP model of most instances with 15 or more customers within 1 hour. The lower bounds for the hard instances, where the MILP did not terminate within 1 hour, are rather small. The large gap is probably caused by the constraints that express implications with heavily weighted indicator variables. This causes the corresponding relaxations to be rather lax.

The results concerning our own Branch-and-Bound algorithm are as follows. On the 17 easy instances, when the MILP was able to verify the optimum trip, the BnB verified it, too. For only one of these instances, it was slightly slower than the MILP, but it was about 10.6 times faster in geometric mean. Furthermore, it verified the exact optimum on 10 more instances within one hour. For 38 of the remaining $72 = 99 - 17 - 10$ instances, BnB gives better upper bounds than the Cplex MILP solver. On the other hand, there are 31 instances for which the MILP gives better upper bounds. We observe that BnB gives worse results for instances, where customers have only few and different opening days and not all customers can be served by a trip.

As already mentioned, an optimal trip was verified for 27 instances (which appear in bold face in the tables), where 10 optimal trips were verified only by the BnB. On 23 of these 27 instances, the optimal trip could

Table 2.2. Relevant parameters and results for real and self-created instances.

Instance	Param.		Optimal costs				Times		
	Cu.	Da.	RI	MILP	BnB	LB	RI	MILP	BnB
brunswick1	12	5	1299.66	**1299.66**	**1299.66**		0.08	1.94	1.33
brunswick2	18	5	1098.27	1098.27	**1098.27**	708.64	0.39	3600	1847
brunswick3	12	5	1239.88	**1239.88**	**1239.88**		0.09	3.34	1.55
brunswick4	17	5	1290.18	1290.18	**1290.18**	845.98	0.33	3600	313
brunswick5	11	5	1257.54	**1257.54**	**1257.54**		0.06	0.80	0.37
brunswick6	18	5	1275.82	1272.46	**1272.46**	764.69	0.36	3600	611
brunswick7	10	5	1176.10	**1176.10**	**1176.10**		0.05	0.12	0.26
brunswick8	16	5	1132.05	1132.05	**1132.05**	826.46	0.26	3600	162
brunswick9	20	5	1210.77	1210.77	3142.18	602.19	0.56	3600	3600
brunswick10	15	5	1285.64	**1285.64**	**1285.64**		0.18	837.10	10.29
rhineland1	32	5	5784.87	5807.45	9861.33	284.46	3.99	3600	3600
rhineland2	32	5	1650.05	1649.78	3647.49	307.67	2.91	3600	3600
rhineland3	31	5	7790.37	7804.70	9849.01	300.21	3.68	3600	3600
rhineland4	36	5	5563.77	5631.14	9742.29	252.99	6.25	3600	3600
rhineland5	32	5	5848.64	5852.85	9938.33	285.66	3.94	3600	3600
rhineland6	32	5	3546.37	3548.82	3641.16	281.46	3.74	3600	3600
rhineland7	28	5	7725.88	7724.93	7845.35	378.56	2.38	3600	3600
rhineland8	37	5	5611.94	5610.49	7725.13	257.81	6.95	3600	3600
baden1	47	5	4184.66	28055.44	35537.62	263.33	16.72	3600	3600
baden2	56	5	10121.23	28211.83	51403.23	278.25	32.07	3600	3600
baden3	55	5	10863.56	23985.45	69037.65	309.92	29.21	3600	3600
baden4	51	5	2457.98	16188.11	47226.46	204.44	18.21	3600	3600
baden5	45	5	3999.03	9988.24	55266.24	290.92	14.24	3600	3600
baden6	56	5	11854.83	26081.42	55425.96	233.85	30.98	3600	3600
baden7	49	5	5943.43	25807.29	27620.70	298.72	21.11	3600	3600
baden8	57	5	11879.94	30202.90	65564.49	189.86	34.41	3600	3600
baden9	45	5	1903.77	3824.54	45344.23	227.56	10.21	3600	3600
baden10	36	4	3817.70	11480.32	23275.06	184.51	6.95	3600	3600
restaurant1	5	2	335.35	**335.35**	**335.35**		0.01	0.97	0.40
restaurant2	7	2	706.35	**704.37**	**704.37**		0.02	38.06	11.84
restaurant3	9	2	829.72	813.76	**813.76**	408.60	0.04	3600	731
restaurant4	10	2	882.63	882.63	882.63	332.45	0.06	3600	3600
restaurant5	13	2	1084.72	1099.29	1086.85	349.10	0.14	3600	3600
germany1	16	5	2930.02	2930.02	3047.85	1419.41	0.32	3600	3600
germany2	8	5	2094.74	2094.74	**2094.74**	1512.76	0.04	3600	314
kiel	7	5	302.29	**302.29**	**302.29**		0.03	1125.91	97.17

Table 2.3. Relevant parameters and results for small random instances.

Instance	Param.			Optimal costs				Times		
	Cu.	Da.	km	RI	MILP	BnB	LB	RI	MILP	BnB
rand1	5	3	100	426.62	**426.62**	**426.62**		0.01	9.56	1.01
rand2	5	3	300	1154.97	**1154.97**	**1154.97**		0.01	3.35	0.15
rand3	5	3	500	3340.08	**3340.08**	**3340.08**		0.01	4.74	0.12
rand4	5	5	100	294.84	**294.84**	**294.84**		0.01	139.65	7.21
rand5	5	5	300	858.86	**858.86**	**858.86**		0.01	168.83	6.84
rand6	5	5	500	1415.62	**1415.62**	**1415.62**		0.01	51.49	0.86
rand7	5	7	100	368.37	**368.37**	**368.37**		0.01	345.21	17.10
rand8	5	7	300	910.27	**910.27**	**910.27**		0.01	570.24	9.66
rand9	5	7	500	1549.42	**1549.42**	**1549.42**		0.01	534.51	12.84
rand10	10	3	100	533.79	533.79	543.71	185.98	0.08	3600	3600
rand11	10	3	300	1158.79	1158.79	**1158.79**	552.27	0.06	3600	367
rand12	10	3	500	11360.94	13264.52	**11360.94**	1037.55	0.06	3600	2
rand13	10	5	100	564.17	564.01	564.17	155.03	0.09	3600	3600
rand14	10	5	300	1125.74	1125.74	1126.15	414.07	0.09	3600	3600
rand15	10	5	500	1733.47	1733.47	**1733.47**	954.72	0.07	3600	368
rand16	10	7	100	532.46	532.46	558.38	186.79	0.10	3600	3600
rand17	10	7	300	1214.39	1211.05	1211.04	521.32	0.10	3600	3600
rand18	10	7	500	1875.01	1875.01	1875.01	826.82	0.08	3600	3600
rand19	15	3	100	778.40	790.58	842.60	203.24	0.30	3600	3600
rand20	15	3	300	7116.08	10184.73	6223.91	539.06	0.33	3600	3600
rand21	15	3	500	9216.67	12165.22	**8371.80**	777.54	0.24	3600	865
rand22	15	5	100	714.55	732.57	869.34	151.70	0.38	3600	3600
rand23	15	5	300	1595.80	1584.40	1584.40	586.27	0.31	3600	3600
rand24	15	5	500	8571.35	13266.42	9074.86	1008.64	0.35	3600	3600
rand25	15	7	100	707.67	667.18	772.74	171.87	0.41	3600	3600
rand26	15	7	300	1467.88	1488.19	1513.61	543.92	0.39	3600	3600
rand27	15	7	500	3342.27	14733.75	2844.34	1007.68	0.26	3600	3600
rand28	20	3	100	926.19	1942.16	1001.45	196.96	0.78	3600	3600
rand29	20	3	300	10729.13	18178.02	14236.97	615.75	0.64	3600	3600
rand30	20	3	500	14819.58	20361.16	14261.57	778.66	0.54	3600	3600
rand31	20	5	100	904.25	941.31	1145.55	171.15	1.01	3600	3600
rand32	20	5	300	1707.85	10849.94	1808.92	472.59	0.78	3600	3600
rand33	20	5	500	6494.18	12102.70	8205.38	983.87	1.08	3600	3600
rand34	20	7	100	870.72	907.23	974.46	198.32	1.09	3600	3600
rand35	20	7	300	2014.31	5501.70	2160.53	586.27	0.92	3600	3600
rand36	20	7	500	3324.53	13539.83	2872.09	1001.06	0.82	3600	3600

Table 2.4. Relevant parameters and results for large random instances.

Instance	Param.			Optimal costs				Times		
	Cu.	Da.	km	RI	MILP	BnB	LB	RI	MILP	BnB
rand37	30	3	100	7034.39	16949.77	9991.21	225.08	4.63	3600	3600
rand38	30	3	300	20658.92	31139.15	32139.01	579.26	2.74	3600	3600
rand39	30	3	500	36811.01	45183.04	36251.84	1079.17	1.27	3600	3600
rand40	30	5	100	1163.96	1291.42	1249.01	218.38	4.54	3600	3600
rand41	30	5	300	9383.70	22969.34	11909.15	685.52	4.68	3600	3600
rand42	30	5	500	26324.49	39065.96	27123.78	1048.96	3.46	3600	3600
rand43	30	7	100	1424.85	1582.90	1552.39	231.65	4.82	3600	3600
rand44	30	7	300	3468.30	16795.94	6839.07	683.70	3.68	3600	3600
rand45	30	7	500	12417.07	40430.71	18041.16	1210.25	5.22	3600	3600
rand46	40	3	100	19027.49	36056.00	28967.30	262.86	11.43	3600	3600
rand47	40	3	300	41203.02	53255.73	47252.69	795.59	5.36	3600	3600
rand48	40	3	500	51222.93	60319.39	50156.91	1313.08	3.80	3600	3600
rand49	40	5	100	1590.98	13757.03	6683.66	247.77	12.73	3600	3600
rand50	40	5	300	28380.71	53973.23	32910.02	779.97	11.95	3600	3600
rand51	40	5	500	32567.18	51729.08	43033.24	1241.64	7.17	3600	3600
rand52	40	7	100	1768.17	4415.88	1877.92	260.63	13.83	3600	3600
rand53	40	7	300	8100.58	35662.86	12840.61	733.77	18.30	3600	3600
rand54	40	7	500	24323.75	44544.10	25707.99	1176.18	13.09	3600	3600
rand55	50	3	100	45458.84	68063.20	53960.16	258.92	19.39	3600	3600
rand56	50	3	300	56200.00	67187.36	59194.93	826.29	9.97	3600	3600
rand57	50	3	500	77785.52	88087.58	82363.07	1412.40	6.02	3600	3600
rand58	50	5	100	12685.05	30832.51	10646.60	266.49	41.53	3600	3600
rand59	50	5	300	38405.97	70702.29	48937.16	768.63	23.34	3600	3600
rand60	50	5	500	51037.43	73980.61	59919.87	1318.53	14.91	3600	3600
rand61	50	7	100	3211.01	26440.91	5313.53	306.91	51.50	3600	3600
rand62	50	7	300	21163.32	43653.72	23635.75	838.22	41.17	3600	3600
rand63	50	7	500	41792.96	62931.39	44969.36	1428.77	24.14	3600	3600

also be found by the heuristic. Also for the difficult instances, in average the heuristic found better trips in shorter time. The heuristic is bad in such cases where the strategy to choose the earliest possible time window with respect to a given service order fails.

The basics of our EA test framework are as described in Section 2.5. We applied it to all 75 instances with more than 20 customers. The stopping criterion was always a population live time of n^3 milliseconds, where n is the number of customers. We experimental found that a population size of 50 is a good choice as well as 10 matings per iteration. A populations objective value is the cost of the TSPTW solution associated with the best individual (the negative fitness value of the best individual). For each instance and each considered operational parameter setting we normalized the time-dependent convergence data to be in $[0, 1]^2$, where the maximum cost does always correspond to the penalty cost of the empty tour. The algorithmic settings are evaluated by considering the averaged convergence behavior over all instances. Dealing with 1-group populations of size 50, Figure 2.3 shows corresponding plots for different combinations of permutation variation operators. The left hand side of Figure 2.4 is the same for isolated crossover and mutation operators. Guided by these results, we decided to use the combination of order based crossover (OX) and simple inversion mutation (SIM) within our multiple-group population experiments. We consider isolated groups as well as cloning migration (cf., subsection 1.5.5) in every n-th generation. We use 16 groups of size 50 in both cases. The application of selection and variation operators within each group is the same as for the 1-group EAs. The results are shown on the right hand side of Figure 2.4 together with the results of the corresponding one-group EA and the solitary random insert heuristic as benchmarks. As expected, multi-group EAs converge faster than the single-group EA in the beginning. They also provide slightly improved final results in average. We further see that the curves of both multi-group EAs are quite congruent, indicating that the migration between groups does not bring much benefit, here.

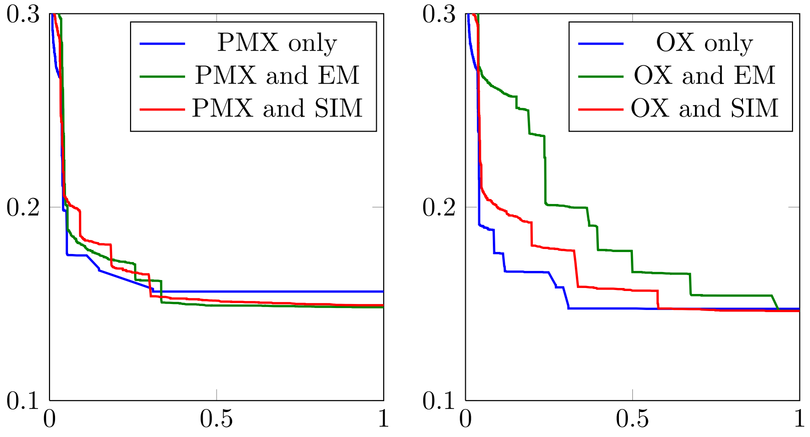


Figure 2.3. Convergence of singe-group EA for different variation operator combinations.

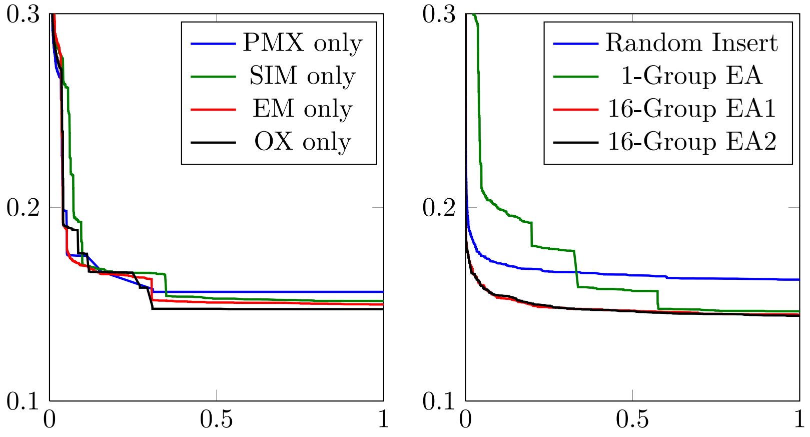


Figure 2.4. Left: Convergence of single-group EA for different variation operators. Right: Convergence of multi-group EA using OX and SIM (EA1 deals with isolated groups, E2 with migration). Benchmarks are given by the corresponding 1-group EA and the solitary random insert heuristic.

Chapter 3

Parameter Optimization of an Ocean Model

3.1 Introduction

The 1-dimensional maritime *biogeochemical (BGC)* model considered here was developed by Schartau and Oschlies [SO03] in 2003. It simulates the interaction of *dissolved inorganic nitrogen (*N*), phytoplankton (*P*), zooplankton (*Z*)* and *detritus (*D*)*. The aim of the *NPZD* model is to reproduce corresponding observations at three North Atlantic locations by the optimization of free parameters within credible limits. In turn, the optimized model can be coupled with 3-dimensional ocean models and help in predicting future oceanic CO_2 uptake, which is supposed to be proportional to the nitrogen concentration.

A FORTRAN implementation of the NPZD model, provided by Ben Ward, NOC Southampton, UK, is currently used within the interdisciplinary research project "Excellence Cluster Future Ocean" at Christian-Albrechts-University, Kiel. The parameter optimization of ocean models is one of the integral parts of the excellence cluster.

Details on the NPZD model and its implementation (which varies from [SO03] in some points concerning the objective function) are given in Section 3.3. A rough mathematical description is as follows.

The interaction of the four tracers is described by source-minus-sink

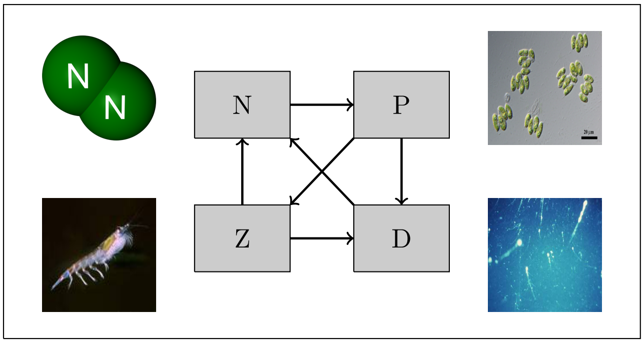


Figure 3.1. The four considered nitrogen states and their interaction in the ocean.

equations

$$\begin{aligned}
\text{sms(N)} &= (-\mu(a) + a_2)\text{P} + a_3\text{Z} + a_5\text{D}, \\
\text{sms(P)} &= (\mu(a) - a_2 - a_6\text{P})\text{P} - \gamma(a)\text{Z}, \\
\text{sms(Z)} &= (a_1\gamma(a) - a_3 - a_4\text{Z})\text{Z}, \\
\text{sms(D)} &= ((1 - a_1)\gamma(a) + a_4\text{Z})\text{Z} + a_6\text{P}^2 - a_5\text{D}
\end{aligned}$$

which correspond to a given time unit and depend on the parameter vector $a \in A \subseteq \mathbb{R}^n$ and somewhat complex functions μ and γ from A into $\mathbb{R}$. The model has between $n = 10$ and $n = 13$ free parameters e. g., we deal with

- a_1 - assimilation efficiency of zooplankton,
- a_2 - linear mortality of phytoplankton,
- ...
- a_{13} - remineralization rate of detritus.

Figure 3.1 sketches the interaction of the four tracers. The 1D-model (66 layers of depth) couples the ocean circulation model *OCCAM* [OCC] with the BGC model using partial differential equations (PDEs)

$$\frac{\partial \text{C}}{\partial t} = -w_{\text{C}}\frac{\partial \text{C}}{\partial z} + \frac{\partial}{\partial z}(K_\rho \frac{\partial \text{C}}{\partial z}) + \text{sms(C)}, \quad \text{C} \in \{\text{N}, \text{P}, \text{Z}, \text{D}\},$$

with sinking velocity parameters w_{C}, turbulent mixing coefficient K_ρ from OCCAM, time t and depth in the ocean z. Actual biweekly measured tracer data $Y_{(\mathrm{C},t,z)}$, $(\mathrm{C},t,z) \in J$, is available from the *Bermuda Atlantic Time-Series (BATS)*. Here, J is the set of all considered tracer-time-depth triples. Accordant interpolation of the model output (1990-1995) leads to a function

$$\varphi : A \mapsto \mathbb{R}^{|J|},$$

where the model output itself is approximatively calculated with sufficient accuracy by finite differences. Finally, we have a weighted *non-linear least squares* optimization (*regression*) problem

$$\min_{a \in A} \quad f(a) := \sum_{(\mathrm{C},t,z) \in J} \omega_{\mathrm{C}} |Y_{(\mathrm{C},t,z)} - \varphi(a)_{(\mathrm{C},t,z)}|^2. \tag{3.1}$$

The parameter domain A is actually given by box constraints, i. e., $A = \prod_{i=1}^{n} [\mathrm{lb}_i, \mathrm{ub}_i] \subseteq \mathbb{R}$ is a product of intervals.

The task is difficult to solve since

- practical optimization of f requires many evaluations of φ, each requiring CPU time in the order of a second,
- the objective function f is non-convex.

Marine biologists carried out efforts to optimize the model with the binary GA framework provided by Carrol [Car] in FORTRAN language but found the results to be unsatisfactorily far from the measured data. For that reason, the task was resumed within the future ocean cluster of excellence, accompanied by the question if the model has to be revised/extended.

Our contribution concerning this matter is twofold. In joined work with the research group *Algorithmic Optimal Control - Oceanic CO$_2$ Uptake* we developed and applied a tool for the parameter optimization problem [RSS+10] (see section Section 3.4).

In general, if no good data-fit is obtained for a parameterized non-linear model, one would like to know some lower bounds on the model error. Inspiration given by problems with (mixed) integer linear programming formulations (like those of Chapter 2 and Chapter 4) is to utilize a kind of relaxation. Under certain circumstances we can use a parameter-free relaxation of the problem type considered here. We address this matter in Section 3.6. The idea is to deal with upper bounds on the smoothness of the objective function. Principally, differential equation based models like that considered here are well suited to give estimations on the smoothness.

3.2 Related Work

There is a lot of work on modeling physics and ecology of the ocean. We only refer to [SO03] and the references there as an entry point.

The model function φ that appears within our box constrained non-linear least squares problem (3.1) is an approximation of the solution of the given PDEs. The corresponding FORTRAN code provided by the maritime biologists does this approximation using a forward Euler scheme with adequate step size. We use the provided code without changes. However, faster evaluations of the objective function may be possible without loss of accuracy, if other time step schemes (e. g., of Runge-Kutta type) are used. A further approach to reduce evaluation time within optimization is proposed by Pries, Slawig, Koziel [PKS11] and Pries, Slawig [PS12].

The optimization tool which we use within our EA framework is an implementation of the *sequential quadratic programming (SQP) approach* and suited for general constrained non-linear optimization problems. Methods for local optimization of non-linear least squares problems are introduced in pertinent monographs like [GMW81][Kos93][OR00][BV04].

3.3 Details on the NPZD Model

3.3.1 Model state variables and parameters

The time-dependent concentrations of the four state variables, N, P, Z and D, are calculated within a 66 level vertical grid of increasingly thick layers. The top 20 layers of the grid are those for which measurements are available. The parameters that influence the interaction of the state variables are listed in Table 3.1.

3.3.2 Data

Currently assimilated data are from the Bermuda Atlantic Time-Series Study (BATS; 31N 64W), i. e., one location only. Other sources of data such

Table 3.1. Model parameters with their physical units, mathematical symbols, traditional values and credible limits

	units	symbol	default	range
fixed parameters				
growth coefficient	1	C_{ref}	1.066	-
growth coefficient	$\frac{1}{^\circ\text{C}}$	c	1	-
PAR extinction length	m	k_w	25	-
short-wave PAR fraction	1	f_{PAR}	0.43	-
tunable parameters				
zooplankton assimilation efficiency	1	β	0.75	0.3 - 0.93
phytoplankton growth rate parameter	$\frac{1}{\text{day}}$	μ_m	0.6	0.2 - 1.46
slope of photosynthesis vs light intensity	$\frac{\text{m}^2}{\text{Wday}}$	α	0.025	0.001 - 0.253
zooplankton loss rate parameter	$\frac{1}{\text{day}}$	Φ_m^z	0.01	0 - 0.63
phytoplankton light attenuation	$\frac{\text{m}^2}{\text{mmol(N)}}$	κ	0.03	0.01 - 0.073
grazing encounter rate	$\frac{\text{m}^6}{(\text{mmol(N)})^2\text{day}}$	ϵ	1	0.025 - 1.6
maximum grazing rate	$\frac{1}{\text{day}}$	g	2	0.04 - 2.56
phytoplankton linear mortality	$\frac{1}{\text{day}}$	Φ_m^p	0.014	0 - 0.63
zooplankton quadratic mortality	$\frac{\text{m}^3}{\text{mmol(N)day}}$	Φ_z^*	0.205	0.01 - 0.955
detritus remineralisation rate	$\frac{1}{\text{day}}$	γ_m	0.02	0.02 - 0.146
half saturation for NO_3 uptake	$\frac{\text{mmol(N)}}{\text{m}^3}$	k_N	0.5	0.1 - 0.73
detritus sinking velocity	$\frac{1}{\text{day}}$	w_s	6	2 - 128
phytoplankton quadratic mortality	$\frac{\text{m}^3}{\text{mmol(N)day}}$	Φ_p^*	0.05	0.01 - 0.955

as the North Atlantic Bloom Experiment (NABE; 47N 20W) and Ocean Weather Ship-India (OWS-INDIA; 59N 19W) are not considered at this time.

Measured ecological BATS data corresponding to the biogeochemical model are available for dissolved inorganic nitrogen (DIN) (mmol(N)m^{-3}), chlorophyll a (Chl a) (mg(Chl a)m^{-3}), zooplankton biomass (ZOO) (mmol(N) m^{-3}), particulate organic nitrogen (PON) (mmol(N)m^{-3}), and C-primary production (CPP) (mmol(C)m^{-3}day^{-1}). Measurements for each of these five tracers were done two times in a month. For each measurement, stored values are the year of observation, the day of the year (one decimal place) the depth (m), and the corresponding value of the tracer. Except for zooplankton biomass, the tracer value is its concentration at the corresponding depth. For zooplankton biomass, the tracer value is the integrated concentration in the water column from the given depth (approximately 200 meters) up to the ocean surface (zooplankton is measured by dragging a 2μm net from depth to the surface). The other tracers were measured at 10 to 15 different depths each time. The depths of measurements variate and do not correspond to the depths in the vertical grid of the NPZD model. The model further requires an initial DIN concentration profile with respect to the vertical grid. This is calculated as the mean depth profile derived from the gridded DIN observations.

The BGC model is forced by output from the OCCAM global circulation model [OCC]. These data estimate the surface irradiance (Wm^{-2}) and the profiles of temperature (°C), salinity (dimensionless), vertical velocity (m s^{-1}), and vertical diffusivity (m s^{-2}), respectively. Hourly OCCAM data is given for every year of the study (1990-1999 inclusive) and (except for irradiance) for the 66 layers of depth of the NPZD model.

3.3.3 NPZD-equations

The (mmol N m^{-3}) concentrations of dissolved inorganic nitrogen, phytoplankton, zooplankton, and detritus are noted by N, P, Z, and D, respectively. We use auxiliary variables with the same mathematical symbols as in [SO03]. Those are stated in Table 3.2.

Now, the biogeochemical source minus sink equations of the four tracers

Table 3.2. Auxiliary variables of the NPZD model

symbol	equation	meaning
z		depth in water column
T		temperature
V_P	$= \mu_m \cdot (C_{ref})^{cT}$	maximum growth rate of phytoplankton
$\gamma(T)$	$= \gamma_m \cdot (C_{ref})^{cT}$	temperature dependent remineralization rate
$\Phi_z(T)$	$= \Phi_m^z \cdot (C_{ref})^{cT}$	temperature dependent phytoplankton growth rate
$\Phi_p(T)$	$= \Phi_m^p \cdot (C_{ref})^{cT}$	temperature dependent zooplankton growth rate
u	$= \frac{\mathrm{N}}{k_\mathrm{N}+\mathrm{N}}$	factor for nutrient limited growth rate of phytoplankton
$\overline{\mu}(z)$		light limited growth rate of phytoplankton, according to Evans and Parslow [EP85], see 3.3.6
$J(\mu, u)$	$= \min(\overline{\mu}(z), V_p)$	growth rate of phytoplankton after Liebig's Law of the Minimum
$G(\epsilon, g)$	$= \frac{g\epsilon P^2}{g+\epsilon P^2}$	zooplankton grazing function

after [SO03] are given by

$$\begin{aligned}
\mathrm{sms(N)} &= [-J(\mu,u) + \Phi_p(T)]\mathrm{P} + \Phi_z(T)\mathrm{Z} + \gamma(T)\mathrm{D}, \\
\mathrm{sms(P)} &= [J(\mu,u) - \Phi_p(T) - \Phi_p^*\mathrm{P}]\mathrm{P} - G(\epsilon,g)\cdot\mathrm{Z}, \\
\mathrm{sms(Z)} &= [\beta G(\epsilon,g) - \Phi_z(T) - \Phi_z^*\mathrm{Z}]\mathrm{Z}, \\
\mathrm{sms(D)} &= [(1-\beta)G(\epsilon,g) + \Phi_z^*\mathrm{Z}]\mathrm{Z} + \Phi_p^*\mathrm{P}^2 - \gamma(T)\mathrm{D}.
\end{aligned} \tag{3.2}$$

The above equations are an extension of the model of Oschlies and Garcon [OG99], where the quadratic loss of phytoplankton is excluded and the linear loss of phytoplankton goes to detritus:

$$\begin{aligned}
\mathrm{sms(N)} &= -J(\mu,u)\mathrm{P} + \gamma(T)\mathrm{D} + \Phi_z(T)\mathrm{Z}, \\
\mathrm{sms(P)} &= [J(\mu,u) - \Phi_p(T)]\mathrm{P} - G(\epsilon,g)\cdot\mathrm{Z}, \\
\mathrm{sms(Z)} &= [\beta G(\epsilon,g) - \Phi_z(T) - \Phi_z^*\mathrm{Z}]\mathrm{Z}, \\
\mathrm{sms(D)} &= [(1-\beta)G(\epsilon,g) + \Phi_z^*\mathrm{Z}]\mathrm{Z} + \Phi_p(T)\mathrm{P} - \gamma(T)\mathrm{D}.
\end{aligned} \tag{3.3}$$

Strictly speaking, the model of [OG99] does not use the values $\gamma(T)$, $\Phi_z(T)$ and $\Phi_p(T)$ but simply their temperature invariant counterparts γ_m, Φ_m^z and Φ_m^p in (3.3). The implementation provides the option (compiler flag) to use either (3.2) or (3.3), but both with the temperature invariant values γ_m, Φ_m^z and Φ_m^p instead of $\gamma(T)$, $\Phi_z(T)$ and $\Phi_p(T)$.

The source minus sink equations are embedded into differential equations of the form

$$\frac{\partial \mathrm{C}}{\partial t} = -w_{\mathrm{C}}\frac{\partial \mathrm{C}}{\partial z} + \frac{\partial}{\partial z}(K_\rho \frac{\partial \mathrm{C}}{\partial z}) + \mathrm{sms}(\mathrm{C}), \tag{3.4}$$

where $\mathrm{C} \in \{\mathrm{N}, \mathrm{P}, \mathrm{Z}, \mathrm{D}\}$, K_ρ is the turbulent mixing coefficient and w_{C} the sinking velocity, which becomes nonzero only for D.

The carbon-primary production is given by

$$\mathrm{PP} = J(\mu, u) \cdot \mathrm{P} \cdot R.$$

3.3.4 Modeloutput

Hourly model output profiles are calculated as follows. At the beginning, the initial profile is used for N, and the components of the other profiles $(\mathrm{P}, \mathrm{Z}, \mathrm{D}, \mathrm{PP})$ are set to small values. The profiles of the next hour are calculated by using the source minus sink equations (3.2)(3.3) four times (the discrete time resolution of the BGC processes is 15 minutes) and the drift and diffusion equation (3.4) once. The procedure is iterated for every hour of the whole considered time period.

3.3.5 Costfunction

Let $\mathrm{C} \in \{\mathrm{N}, \mathrm{P}, \mathrm{Z}, \mathrm{D}, \mathrm{PP}\}$ be a tracer, a be a runtime year and $N(\mathrm{C}, a)$ denote the number of measurements of tracer C in the year a. For any $i \in N(\mathrm{C}, a)$ the i-th measurement of tracer C in the year a is written as $y(\mathrm{C}, a, i)$ and the corresponding time and depth as $t(\mathrm{C}, a, i)$ and $z(\mathrm{C}, a, i)$, respectively.

The depth of the middle of the k-th model layer, $1 \leqslant k \leqslant 66$, is denoted by $z_m(k)$. We further set $z_m(0) = 0$. The nearest upper and lower layers to $z(\mathrm{C}, a, i)$ (with respect to their centers) are determined as

$$\begin{aligned} \overline{k}_{\mathrm{C},a,i} &= \max\{0 \leqslant k \leqslant 66 \mid z_m(k) \leqslant z(\mathrm{C}, a, i)\} \\ \underline{k}_{\mathrm{C},a,i} &= \min\{1 \leqslant k \leqslant 66 \mid z_m(k) \geqslant z(\mathrm{C}, a, i)\} \end{aligned}$$

We define $\tilde{f}(\mathrm{PP}, a, h, k)$ as the modeled carbon-primary production in the 24 hours time interval $[h-12,\ h+11]$ around the h-th hour of the year a in layer k, whereas $\tilde{f}(\mathrm{C}, a, h, k)$ is only the corresponding single model output value, if $\mathrm{C} \neq \mathrm{PP}$. We further set $\tilde{f}(\mathrm{C}, a, h, 0) = \tilde{f}(\mathrm{C}, a, h, 1)$ for every C, a and h.

Now, for every measurement $y(\mathrm{C}, a, i)$ a corresponding value $f(\mathrm{C}, a, i)$ can be derived from the model output. Booth values are scaled with respect to the same measuring unit. For $\mathrm{C} \neq \mathrm{Z}$, the linear interpolation

$$f(\mathrm{C}, a, i) = \alpha \cdot \tilde{f}(\mathrm{C}, a, h_{\mathrm{C},a,i}, \overline{k}_{\mathrm{C},a,i}) + (1-\alpha) \cdot \tilde{f}(\mathrm{C}, a, h_{\mathrm{C},a,i}, \underline{k}_{\mathrm{C},a,i}),$$

with $h_{\mathrm{C},a,i} = \lfloor t(\mathrm{C}, a, i) \rfloor$ and $\alpha = \frac{z_m(k_2) - z(\mathrm{C},a,i)}{z_m(k_2) - z_m(k_1)}$ is taken, whereas $f(\mathrm{Z}, a, i)$ is $\frac{1}{z(\mathrm{C},a,i)}$ times the integral of the accordant piecewise linear function over the water column $[0, z(\mathrm{C}, a, i)]$.

Now, the over all cost function is calculated as

$$\frac{1}{\nu} \sum_{a \in A} \sum_{\mathrm{C} \in \mathrm{Tr}} \frac{1}{\sigma_{\mathrm{C}} \cdot N(\mathrm{C}, a)} \sum_{i=1}^{N(\mathrm{C},a)} (f(\mathrm{C}, a, i) - y(\mathrm{C}, a, i))^2,$$

where A is the set of model runtime years except for the first year (spinup year), $\mathrm{Tr} = \{\mathrm{N}, \mathrm{P}, \mathrm{Z}, \mathrm{D}, \mathrm{PP}\}$ is the set of the tracers, ν is the cardinality of $\{(\mathrm{C}, a) \mid \mathrm{C} \in \mathrm{Tr}, a \in A, N(\mathrm{C}, a) \neq 0\}$ and σ weights the different tracers and is currently set to 0.1 for N, 0.01 for P, 0.01 for Z, 0.0357 for D and 0.025 for PP.

3.3.6 Growth of Phytoplankton

The source minus sink equations of the NPZD model are effected by the light limited growth rate $\mu(z, t)$ of phytoplankton which varies with depth z and time t. Average light limited phytoplankton growth rates $\overline{\mu(k, t)}$ are calculated for every layer k of depth using a simplified version of an approximative formula by Evans and Parslow [EP85].
The basic formula for the light limited growth rate as a function of depth and time uses the curve of Smith, as recommended by Jassby and Platt when analytic integration is desired. The curve of Smith is applied to the

variable irradiance:

$$\mu(z,t) = \frac{V_{\mathrm{P}} \cdot \alpha \cdot I(z,t)}{\sqrt{V_{\mathrm{P}}^2 + (\alpha \cdot I(z,t))^2}}$$

where V_{P} is the maximum growth rate of phytoplankton, α is the initial slope of photosynthesis vs light intensity, and $I(z,t)$ is the solar irradiance at depth z and time t. Evans and Parslow deal with a triangular approximation of the daily curse of the sun leading to a double integral for the representation of average phytoplankton growth rate within one layer of depth.
In our case, time dependent values of the solar irradiance $I(0,t)$ at the ocean surface are taken from the physical model, and the solar incidence angle β_{air} at noon is assumed to be the equivalent daily averaged incidence angle for direct and diffuse radiation. The cosine of the solar incidence angle β_{water} in water corresponds to the relative way of light per depth and is calculated after Snells law $[\sin(\beta_{water}) = \sin(\beta_{air}/1.33)]$:

$$\cos(\beta_{water}) = \sqrt{1 - \frac{1-\cos^2(\beta_{air})}{1.33^2}}.$$

The light attenuation factor per depth is supposed to be caused by water and phytoplankton only. For the k-th grid box, it is calculated as

$$\kappa(k,t) = \frac{1}{\cos(\beta_{water}) \cdot k_w} + \frac{\kappa \cdot \mathrm{P}(k,t)}{\cos(\beta_{water})}.$$

With z_k and $z_k + 1$ as the top of the k-th and $(k+1)$-th box, respectively, we obtain

$$\overline{\mu(k,t)} = \frac{V_{\mathrm{P}}}{z_{k+1} - z_k} \int_{z=0}^{z_{k+1}-z_k} \frac{\alpha \cdot I(z_k,t) \cdot e^{-\kappa(k,t)z}}{\sqrt{V_{\mathrm{P}}^2 + (\alpha \cdot I(z_k,t) \cdot e^{-\kappa(k,t)z})^2}} dz,$$

where the irradiance $I(z_k,t)$ at the top of the k-th box is given by

$$I(z_k,t) = I(0,t) \cdot e^{-\sum_{j=2}^{k} \kappa(j,t)(z_j - z_{j-1})}$$

Substitution of depth z by light intensity $\varphi(z) = \alpha \cdot I(z_k,t) \cdot e^{-\kappa(k,t)z}$ gives

$$\overline{\mu(k,t)} = -\frac{V_{\mathrm{P}}}{\kappa(k,t)(z_{k+1} - z_k)} \int_{y=\varphi(0)}^{\varphi(z_{k+1}-z_k)} \frac{1}{\sqrt{V_{\mathrm{P}}^2 + y^2}} dy$$

with analytical solution

$$\overline{\mu(k,t)} = \frac{V_\mathrm{P}}{\kappa(k,t)(z_{k+1}-z_k)} \cdot \ln\left(\frac{\varphi(0)+\sqrt{V_\mathrm{P}^2+\varphi(0)^2}}{\varphi(z_{k+1}-z_k)+\sqrt{V_\mathrm{P}^2+\varphi(z_{k+1}-z_k)^2}}\right).$$

3.4 EA Framework

In joint work with the research group *Algorithmic Optimal Control - CO$_2$ Uptake of the Ocean* of Thomas Slawig we developed an optimization framework for the model that can also be applied to other parameter optimization problems. The article [RSS$^+$10] concerns experimental studies using the tool and confirms conjectures of the marine biologists about the NPZD model. Our framework hybridizes genetic operators with deterministic gradient based search. Base is the simple classical EA framework as described in Section 1.5. The set of genotypes is $\mathcal{G} = [0,1]^n$, i. e., we operate on normalized parameter values. The selection operator is linear ranking based roulette-wheel selection with pressure parameter 2 (see subsection 1.5.3). Our variation operators are BLX-alpha crossover, which is well suited if no preferences for offspring properties are given [YG10], and two mutation operators. The first mutation operator performs single gradient based search steps, supporting the exploitation ability of the algorithm. For this purpose, we use the C software library CFSQP, an implementation of the *sequential quadratic programming (SQP)* method. Having the option to provide first order derivations of the fitness function to be passed to CFSQP, we used *automatic differentiation (AD)* to generate corresponding code. This yields both less computational effort and higher numerical stability of the SQP steps. Since experiments will show frequent convergence to solutions with the same unsatisfying objective value, we also use uniform mutation (i. e., $x_i = U([0,1])$, $i \in [n]$) in order to allow more exploration of the search space. Here, a mutant $\tilde{x}$ of an individual x is obtained by choosing some random alleles to become $\tilde{x}_i = U(I_i)$, where the interval I_i is either $[0,1]$ (global search) or $[\min(x_i, x_i^\mathrm{best}), \max(x_i, x_i^\mathrm{best})]$ with currently fittest individual x^best (increasingly local search). The probabilities for the choice of I_i can be fix or tend from the first alternative in the beginning toward the second alternative in the end of the algorithm running time. The latter realization of uniform mutation corresponds to the quantum inspired evolutionary algorithm operation for real coded problems that is given in [BDP09].

Like in the former chapter, we parallelize the framework by subdivision of the population into groups of equal size.

3.5 Experiments

Again, our experiments are carried out on a Linux system with AMD Opteron CPUs having 2100 MHz clock rate, 512 kB cache and 32 GB of RAM. Our framework allows entirely gradient based search with random start solutions (using many individuals, switching off crossover and using SQP mutation). Starting with this setting, we observe convergence of many individuals (roughly one third of the population) to the fitness value that marine biologist obtained before. The fact that this ratio is true for both, the actual measured data from BATS (yielding an unsatisfying least squares error norm) and former generated model output by a certain parameter vector (yielding error norm ≈ 0) does give some evidence to the conjecture that the bad data-fit of the BATS measurements is already best possible w.r.t. the model. The error norm of this data-fit is about 40% of the associated norm of the measured data itself.

Concerning the BATS measurements, we further observed a wide parameter spread under those solutions the fitness value of which is close to that of the best solution a^*. We illustrate this fact by Figure 3.2. Here, we first optimized each of 100000 individuals by 10 SQP steps and let the 1000 fittest survive. Their fitness values are within 25% of $f(a^*)$, but their genes are very widely spread. Applying 100 further SQP steps to each of them and selecting the fittest 250 individuals, the fitness values are distributed within 1% of $f(a^*)$ but many genes show still a quite large variation. At least, we could check that the final optimization result (after termination of the SQP procedure) is a KKT point, which for our problem means exactly what one does intuitively expect:

$$\nabla f(a)_i \begin{cases} = 0 & \text{if } a_i \in (\mathrm{lb}_i, \mathrm{ub}_i) \\ \leqslant 0 & \text{if } a_i = \mathrm{u}_i \\ \geqslant 0 & \text{if } a_i = \mathrm{l}_i \end{cases}$$

Our remaining optimization attempts are done as follows. We use a parallel

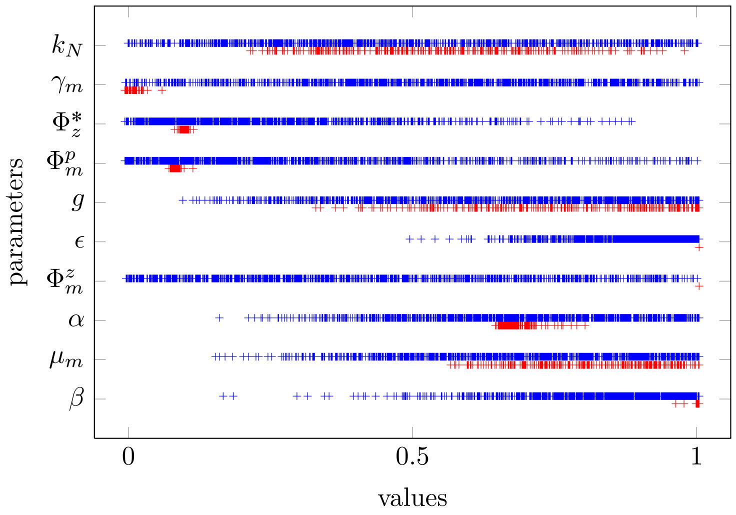


Figure 3.2. Distribution of the components (cf., Section 3.3) of the best 1000 normalized parameter vectors of the NPZD model after optimizing each of 100000 random start parameter vectors with 10 SQP steps (blue marks) and the best 250 parameter vectors after optimizing each of the 1000 parameter vectors with 100 further SQP steps (red marks).

implementation with 64 independent groups of size 10. As we seek for a local optimal solution that is better than the frequently reached best known solution, we abandon migration for exploration reasons, here. For the same reason, we use the full interval $[0, 1]$ for uniform mutations and try uniform mutation only as well as the combination of uniform mutation with SQP mutation. Both mutation variants were testet with BLX-alpha crossover and without crossover, respectively. In addition to a random start population that is generated using uniform distribution w.r.t. the whole parameter domain, we also tried a start with an initial population that was given by a set of pareto optimal solutions. Namely, we used the Euklidian distance of individuals to the best known solution x^* as second fitness function (in addition to the negated least squares error norm f of the associated model output). The set of pareto optimal solutions (the set of solutions which are not inferior to other solutions w.r.t. both objectives) that was obtained by running the EA framework regarding both these objectives was chosen to be our alternative initial population. Despite our observations concerning the parameter spread of good individuals, our hope was to find better optima than x^* by starting with individuals that provide good data-fit errors but are also quite far away from x^*.

In contrast to Chapter 2 and Chapter 5, we refrain from comparing the convergence behavior for different operational parameter settings since not much conclusions can be drawn regarding the special situation of frequent convergence to the same optimum value, here.

None of our optimization attempts yielded solutions that are superior to x^*, which gives more evidence that the considered NPZD model has to be extended in order to meet the assimilated data properly.

3.6 Lower Bounds for Non-Linear Regression

Like for the NPZD model, the task to simulate physical, biological or chemical processes is often a matter of regression problems. In this connection, parameters of suitable model functions have to be adapted such that experimental data is fit as good as possible. Depending on the chosen error norm, efficient computation of globally optimal parameter sets is possible for certain model functions like linear functions or polynomials. But in many cases, the functions that are supposed to mirror reality are non-linear and non-convex. In such cases, only local optima may be obtained. Now, if the obtained solutions are not satisfying (as for the NPZD model), one is interested to know if there are satisfying model parameters at all. A negative answer to this question would be given by a theoretical lower error norm bound which is greater than desired. This would imply that either the experimental data or the model is invalid. One reason for a model to be improper to allow any good data-fit may be that there exist higher frequency/amplitude processes as incorporated by the model. Concerning this aspect, we give lower bounds on the least squares error norm under the assumption that limits on the derivations of the model function are known.

3.6.1 The Regression Problem

We consider the regression problem

$$\min_{a \in A} f(a) := \sum_{i=1}^{N} |\varphi(a; x_i) - y_i|^2, \tag{3.5}$$

where $A \subseteq \mathbb{R}^n$ is a set of free parameters, $x_1 < x_2 < \cdots < x_N$, are independent sampling points and y_i, $i \in [N]$ accordant measured data, roughly supposed to depend on x as described by the parameterized model function $\varphi : A \times \mathbb{R} \to \mathbb{R}$.

3.1 Remark. We suppose φ to be given analytically. In practice, φ is often given implicitly, usually in terms of differential equations which have to be solved by numerical methods causing some approximation error.

Let $k \in \mathbb{N}_0$. We assume that $\varphi(a; \cdot)$ is $k+1$ times continuously differentiable and $|\varphi(a; \cdot)^{(k+1)}| \leqslant D$ for every $a \in A$. We aim to find a maximum $\alpha \in \mathbb{R}_{>0}$ with

$$\alpha \leqslant f(a) \quad \text{for all } a \in A. \tag{3.6}$$

Actually, although dealing with parameterized models in practice, our lower bound is applicable to any $k+1$ times continuously differentiable function $\phi : \mathbb{R} \to \mathbb{R}$ with $|\phi^{(k+1)}| \leqslant D$, $D \in \mathbb{R}_{>0}$. We denote the set of these function with $C^{k+1}(\mathbb{R}, \mathbb{R}, D)$. Since we assumed for all $a \in A$ that the functions $\varphi(a; .)$ in (3.5) are in $C^{k+1}(\mathbb{R}, \mathbb{R}, D)$, the task

$$\min_{\phi \in C^{k+1}(\mathbb{R},\mathbb{R},D)} \quad \sum_{i=1}^{N} |\phi(x_i) - y_i|^2 \tag{3.7}$$

is a relaxation of (3.5).

3.6.2 The 1-Norm Case

To illustrate the simple idea, let us first consider the $\|\cdot\|_1$ norm. Figure 3.3 shows the plot of some experimental data and the graph of a model function ϕ. We suppose that $\phi \in C^1(\mathbb{R}, \mathbb{R}, D)$ for some $D \in \mathbb{R}_{>0}$. Thus, for some fix $\overline{x} \in \mathbb{R}$ and the sampling points $x_i \in \mathbb{R}$, $i \in [N]$ the difference between $\phi(x_i)$ and the zeroth Taylor polynomial of ϕ around $\overline{x}$ (being the constant function with value $\phi(\overline{x})$) is upper bounded by $D|x_i - \overline{x}|$. The light green region in

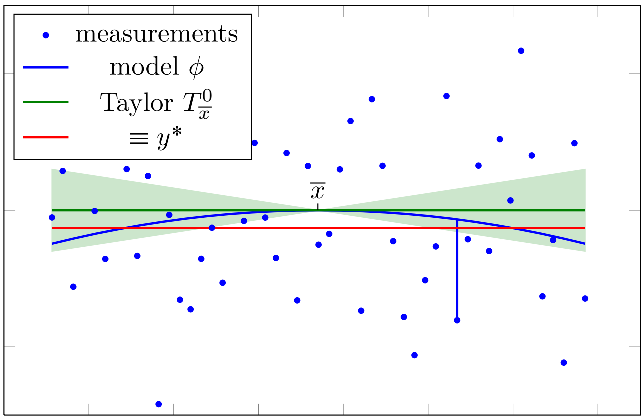


Figure 3.3. Illustration on the construction of a lower bound on a regression problem in the $\|\cdot\|_1$ norm

Figure 3.3 is the possible region of ϕ with respect to a certain $\overline{x}$ and this Taylor bound. It follows by Δ-inequality that the difference between the modeled value of a sampling point and the corresponding measurement y_i satisfies

$$\begin{aligned} |\phi(\overline{x}) - y_i| &\leqslant |\phi(\overline{x}) - \phi(x_i)| + |\phi(x_i) - y_i| \\ &\leqslant D|x_i - \overline{x}| + |\phi(x_i) - y_i| \end{aligned} \tag{3.8}$$

Summation of (3.8) gives

$$\sum_{i=1}^{N} |\phi(\overline{x}) - y_i| \leqslant D \sum_{i=1}^{N} |x_i - \overline{x}| + \sum_{i=1}^{N} |\phi(x_i) - y_i|. \tag{3.9}$$

Now, let $y^* \in \mathbb{R}$ be the best constant $\|\cdot\|_1$-approximation to the data, i. e.,

$$y^* := \operatorname*{argmin}_{y \in \mathbb{R}} \sum_{i=1}^{N} |y_i - y^*|.$$

We may replace the constant $\phi(\overline{x})$ on the left hand side of (3.9) by y^*. Thus, if we set

$$\alpha := \sum_{i=1}^{N} |y_i - y^*| - D\sum_{i=1}^{N} |x_i - \overline{x}|$$

we have

$$\alpha \leqslant \sum_{i=1}^{N} |\phi(x_i) - y_i|.$$

3.2 Remark. Clearly, the lower bound α may be negative if either our general derivation bound D or the distances of the sampling points to $\overline{x}$ are too large. If this is the case there may still be the chance to obtain a positive lower bound by piecewise application of the above procedure. We will get back to this matter when we have discussed the $\|\cdot\|_2$ case.

3.6.3 The 2-Norm Case

Now, let $D \in \mathbb{R}_{>0}$ such that $|\phi^{(k+1)}| \leqslant D$ and $\overline{x} \in \mathbb{R}$. The k-th Taylor polynomial of ϕ in $\overline{x}$

$$T_{\overline{x}}^{k}(x) := \sum_{j=0}^{k} \frac{1}{j!}\phi^{(j)}(\overline{x})(x - \overline{x})^j$$

satisfies

$$|T_{\overline{x}}^{k}(x_i) - \phi(x_i)| \leqslant \frac{D}{(k+1)!}|x_i - \overline{x}|^{k+1} =: M_i$$

which implies

$$\begin{aligned} |T_{\overline{x}}^{k}(x_i) - y_i| &\leqslant |T_{\overline{x}}^{k}(x_i) - \phi(x_i)| + |\phi(x_i) - y_i| \\ &\leqslant M_i + |\phi(x_i) - y_i| \end{aligned}$$

i. e.,

$$|T_{\overline{x}}^{k}(x_i) - y_i| - M_i \leqslant |\phi(x_i) - y_i| \tag{3.10}$$

From the single error bounds (3.10) we can obtain similar bounds concerning the squared differences. If the left hand side of (3.10) is negative, we have

$$|T_{\overline{x}}^{k}(x_i) - y_i|^2 - M_i^2 < 0 \leqslant |\phi(x_i) - y_i|^2$$

Otherwise, we may square both sides and obtain

$$|T^k_{\overline{x}}(x_i) - y_i|^2 - 2 \cdot |T^k_{\overline{x}}(x_i) - y_i| \cdot M_i + M_i^2 \leqslant |\phi(x_i) - y_i|^2$$

In both cases we have

$$|T^k_{\overline{x}}(x_i) - y_i|^2 - 2 \cdot |T^k_{\overline{x}}(x_i) - y_i| \cdot M_i - M_i^2 \leqslant |\phi(x_i) - y_i|^2 \tag{3.11}$$

By summation of (3.11) we obtain

$$\sum_{i=1}^N |\phi(x_i) - y_i|^2 \geqslant \sum_{i=1}^N |T^k_{\overline{x}}(x_i) - y_i|^2 - 2 \cdot \sum_{i=1}^N |T^k_{\overline{x}}(x_i) - y_i| \cdot M_i - \sum_{i=1}^N M_i^2. \tag{3.12}$$

The inequality of Cauchy Schwartz implies that the right hand side of (3.12) is at least

$$\sum_{i=1}^N |T^k_{\overline{x}}(x_i) - y_i|^2 - 2 \cdot \sqrt{\left(\sum_{i=1}^N |T^k_{\overline{x}}(x_i) - y_i|^2\right) \cdot \left(\sum_{i=1}^N M_i^2\right)} - \sum_{i=1}^N M_i^2. \tag{3.13}$$

Let P^* be a regression polynomial of degree k, i. e.,

$$P^* := \operatorname*{argmin}_{P \in \Pi_k} \sum_{i=1}^N |P(x_i) - y_i|^2.$$

The computation of P^* can efficiently be done by solving a system of linear equations. Clearly, P^* is a better approximation to the measurements than the approximation by our Taylor polynomial of degree k, i. e.,

$$\sum_{i=1}^N |P^*(x_i) - y_i|^2 \leqslant \sum_{i=1}^N |T^k_{\overline{x}}(x_i) - y_i|^2. \tag{3.14}$$

Now, setting

$$c := \frac{\sum_{i=1}^N M_i^2}{\sum_{i=1}^N |P^*(x_i) - y_i|^2}$$

we can rewrite (3.13) as

$$\sum_{i=1}^{N} |T_{\overline{x}}^{k}(x_i) - y_i|^2 - 2\sqrt{c} \cdot \sqrt{\left(\sum_{i=1}^{N} |T_{\overline{x}}^{k}(x_i) - y_i|^2\right) \cdot \left(\sum_{i=1}^{N} |P^*(x_i) - y_i|^2\right)} \tag{3.15}$$

$$- c \cdot \sum_{i=1}^{N} |P^*(x_i) - y_i|^2.$$

We can make (3.15) smaller by replacing the second factor under the square root with the right hand side of (3.14) which yields

$$(1 - 2\sqrt{c}) \cdot \sum_{i=1}^{N} |T_{\overline{x}}^{k}(x_i) - y_i|^2 - c \cdot \sum_{i=1}^{N} |P^*(x_i) - y_i|^2. \tag{3.16}$$

If we further assume that $1 - 2\sqrt{c} \geqslant 0$, i. e., $c \leqslant \frac{1}{4}$, we can apply (3.14) conversely and find that (3.16) is at least

$$(1 - 2\sqrt{c} - c) \cdot \sum_{i=1}^{N} |P^*(x_i) - y_i|^2$$

We have proofed the following

3.3 Theorem. *Let $N \in \mathbb{N}$ and $x_i, y_i \in \mathbb{R}$, $i \in [N]$. Let $k \in \mathbb{N}_0$, $D \in \mathbb{R}_{>0}$ and $\phi : \mathbb{R} \to \mathbb{R}$ be any function in $C^{k+1}(\mathbb{R}, \mathbb{R}, D)$. Let $\overline{x} \in \mathbb{R}$ and set*

$$M_i := \frac{D}{(k+1)!} |x_i - \overline{x}|^{k+1}.$$

Let P^* be a regression polynomial of the data pairs $(x_i, y_i) \in \mathbb{R}^2$, $i \in [N]$, i. e.,

$$P^* := \operatorname*{argmin}_{P \in \Pi_k} \sum_{i=1}^{N} |P(x_i) - y_i|^2.$$

Set

$$c := \frac{\sum_{i=1}^{N} M_i^2}{\sum_{i=1}^{N} |P^*(x_i) - y_i|^2}$$

and

$$\alpha := (1 - 2\sqrt{c} - c) \cdot \sum_{i=1}^{N} |P^*(x_i) - y_i|^2.$$

If $c \leqslant \frac{1}{4}$*, it holds that*

$$\alpha \leqslant \sum_{i=1}^{N} |\phi(x_i) - y_i|^2.$$

3.4 Remark. Theorem 3.3 utilizes error bounds on a $k + 1$ times continuous differentiable model function w.r.t. corresponding Taylor polynomial interpolation of degree k. Crucial is only the fact that we have some polynomial of degree k for which such error bounds are known. Another polynomial type that can be utilized is interpolation polynomials through $k+1$ different sampling points $\hat{x}_j$, $j \in [k + 1]$ (not to be confused with the data samples x_i, $i \in [N]$). This provides the more general error bounds

$$M_i := \frac{D}{(k+1)!} \prod_{j=1}^{k+1} (x_i - \hat{x}_j), \quad i \in [N],$$

which are especially good if the $\hat{x}_j$ are chosen as the Chebyshev roots

$$\hat{x}_j = \frac{1}{2}(a + b) + \frac{1}{2}(b - a) \cos\left(\frac{2j - 1}{2n}\pi\right), \quad j \in [k + 1]$$

within the interval $[a, b] = [x_1, x_N]$ of interest (see e. g., [KC09]).

3.5 Remark. Similar to the $\|\cdot\|_1$ case, the derived lower bound α can be negative. This will particulary be true, if the sampling point interval is wide, causing large M_i values. In this case, one may try to apply Theorem 3.3 piecewise by partitioning the interval $[x_1, x_N]$ into narrow subintervals, summing up the obtained lower bounds. As long as there are enough sampling points within the subintervals, the constant c will decrease by an order that depends on the degree of the regression polynomial P^*. The lower bound α will be positive if the ratio c of the summed squared Taylor error bounds and the $\|\cdot\|_2^2$-error of the regression polynomial is roughly less than 1/6. The task will be to find a partition of the sampling point interval such that the sum of the obtained lower bounds is as great as possible.

3.6.4 Application Examples

The applicability of Theorem 3.3 may seem somewhat unclear without examples. We give two simple examples, a self made one and a physical application.

A Simple Harmonic Model

For our first example we simply draw 500 equidistant samples from a sine wave $\sin(x)/2$ on the interval $[0, 2\pi]$ and add some Gaussian noise in order to simulate measurement errors. The obtained data $(x_1, y_1), \ldots, (x_{500}, y_{500})$ is the dots in Figure 3.4. The least squares error of this data w.r.t. the original function is

$$\sqrt{\sum_{i=1}^{500}(y_i - \frac{\sin x_i}{2})^2} \approx 1.058.$$

Considering the plot of samples it is still obvious that the underlying original function was some sine wave. A modeler will use the parameterized function

$$\varphi(a_1, a_2, a_3, a_4; x) = a_1 + a_2 \sin(a_3 x + a_4)$$

and find rough box constraints on the amplitude and the frequency, respectively, say

- $b \in [0.4, 0.6]$
- $c \in [0.9, 1.1]$

i. e., the parameter set is $A = [0.4, 0.6] \times [0.9, 1.1] \times \mathbb{R} \times \mathbb{R}$. Parameter optimization finds an $a^* \in A$ which is approximately the vector $(0, \frac{1}{2}, 1, 0)^T$ of the original (supposed to be unknown) function. The corresponding error norm is ≈ 1.055. If one regards a plot of this solution and the data points, it will be reasonable to conjecture that no much better data fit is possible by a similarly smooth model. For the given setting, we can calculate that the third derivation of the model function satisfies

$$\varphi^{(3)}(x) \leqslant 0.8 \quad \text{for all } a \in A.$$

Now, we can try and apply Theorem 3.3 for functions in $C^3(\mathbb{R}, \mathbb{R}, 0.8)$ using a suitable division of the sampling interval $[0, 2\pi]$. For the division shown in

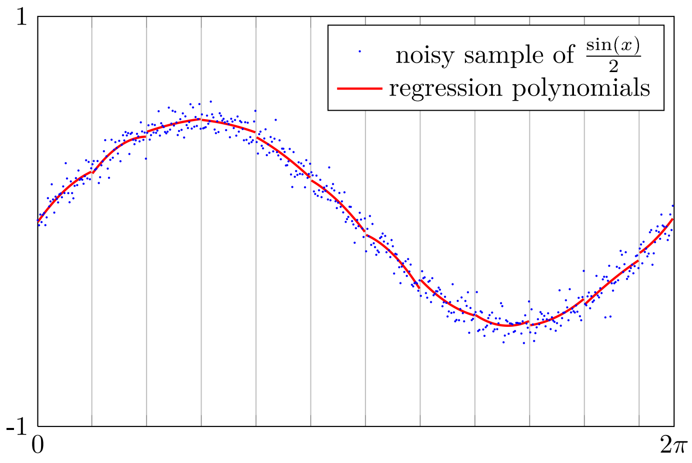


Figure 3.4. Piecewise regression of a noisy sine wave sample by polynomials of degree 2.

Figure 3.4 where the $\overline{x}$ are chosen to be the centers of the subintervals, we calculate the corresponding Taylor bounds as well as regression polynomials of degree 2 and finally derive a lower least squares error norm bound of ≈ 1.014. The bound is within 4% of the corresponding error of the optimized model and, thus, proves the above conjecture.

A Solar UV Radiation Model

We consider a cloud of ultra-violet radiation measurements at clear sky conditions depending on the solar elevation angle x. A simple model supposes the extraterrestrial ultra-violet radiation u_0 to be absorbed by a thin atmospherical layer some h kilometers above the ground having some attenuation factor at for perpendicularly passing light:

$$\mathrm{uv} = \varphi(\mathrm{uv}_0, h, \mathrm{at}; x) = \mathrm{uv}_0 \cdot \mathrm{at}^{-\frac{1}{\sqrt{1-(\frac{r}{r+h}\cdot\cos(x))^2}}}$$

with Earth's radius r. The parameters have constant values and box constraints

$$\begin{aligned} \mathrm{uv}_0 &= 0.6176 \\ r &= 6366 \\ h &\in [10, 500] \\ \mathrm{at} &\in [3, 10] \\ x &\in [0, \tfrac{\pi}{2}]. \end{aligned}$$

If the parameters are within those natural bounds, we can estimate that $|\varphi(\mathrm{uv}_0, h, \mathrm{at}; \cdot)'| \leqslant 0.35$ and $|\varphi(\mathrm{uv}_0, h, \mathrm{at}; \cdot)''| \leqslant 1.2$. We further have that the considered model function is monotonically increasing for every parameter set. For that reason, we can estimate the quality of our bounds by comparison with the best monotonic regression which is a step function and can be efficiently calculated (see, e. g., [BBBB72]). Figure 3.5 shows a cloud of 307nm UV measurements in dependence of the solar elevation angle and the corresponding model function with optimized parameters as well as a plot of the optimum monotonic regression function. Figure 3.6 shows lower bounds obtained by Theorem 3.3 for different equidistant partitions of the sampling interval, using Taylor bounds and regression polynomials of degree 0 and 1, respectively.

The measurements were taken at Sylt Island (Germany) and provided by *Prof. Dr. C. Stick, Institute of Medical Climatology, CAU-Kiel.* The error norm of the parameter optimized model function is 2.19 while monotonic regression has a little bit less error norm of 2.15. The best lower bound by our method using the second derivation estimation is 1.99. From the results we can draw two conclusions. Monotonic regression provides a better lower bound as our method, here. However, no such estimation can be done for models that are not monotonic. The best lower bound by our method is already good enough to state that the parameterized model is a quite suitable choice in the set $C^2(\mathbb{R}, \mathbb{R}, 1.2)$ of all model functions with similar smoothness properties. The second conclusion is that the obtained error bounds confirm the impression that we get from a view of the data cloud, namely that the error of a smooth curve approximation to the data will always be great in relation to the norm of the data vector itself. The reason is, that our model function does not incorporate all parameters on which the ultra-violet radiation depends. An unconsidered but important factor is the density of atmospherical aerosol. Aerosol is subject to variation and,

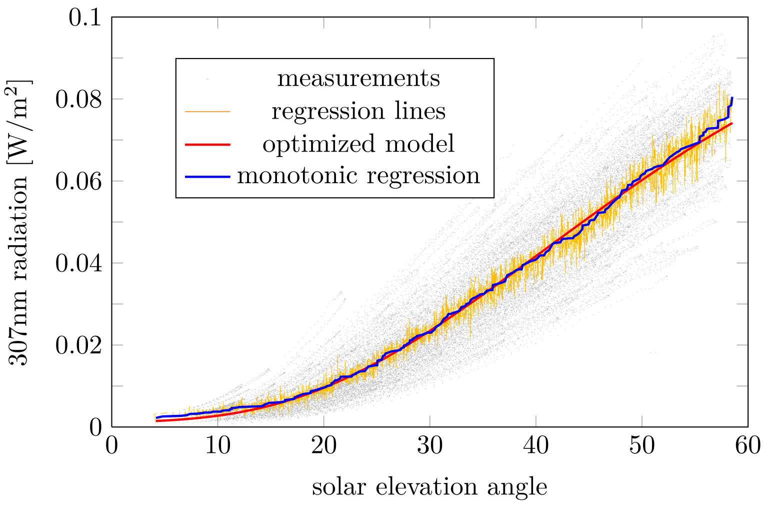


Figure 3.5. UV measurements of 307nm wave at Sylt Island (1995-2004, every 6 minutes at clear sky conditions) and corresponding data fits by lines (piecewise), the parameterized model and a monotonic function

actually, the ozone layer which absorbs most of the ultra-violet radiation is subject to variation, too.

3.6.5 Application Difficulties Concerning the NPZD Model

As already mentioned, differential equation based models are principally suited to give smoothness estimations in order to apply our regression bound procedure. Concerning the biogeochemical ocean model, the variation of the four considered tracers depends on their concentrations and the given vertical drift and diffusion parameters, respectively. Assuming that we have a time series that provides concentration measurements of all tracers and for all depth grid points of the 1-dimensional model, we can proceed as follows. In a first step we determine the maximum measured concentration for each tracer and each layer of depth within either the whole time horizon

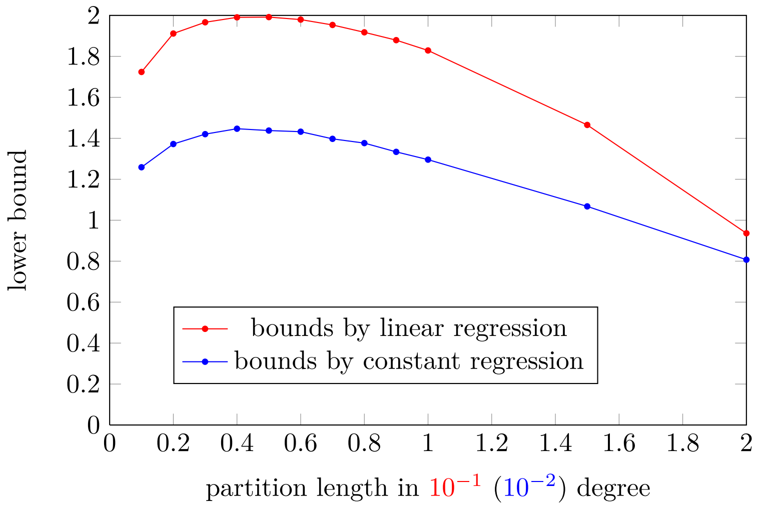


Figure 3.6. Lower regression bounds on the UV model using different equidistant subdivisions of the sampling interval

or within predefined subintervals of the time horizon. We decide to be unsatisfied with our model if it exceeds those maximum values by more than a certain factor, say 2. Thus, using twice the maximum measured concentrations, we can separately estimate the maximum model variation for each tracer and each layer of depth, incorporating the box constraints on the free parameters. Finally, we sum all corresponding lower error norm bounds to obtain the lower bound on the error norm of the ocean model.

Unfortunately, our assumption about the time series is not completely fulfilled. As described in Section 3.3, measurements of dissolved nitrogen, phytoplankton and detritus have been taken at varying depths which do not correspond to the depth grid points of the ocean model. The measurements have to be interpolated to meet the model depth grid. Even worse is the situation concerning zooplankton dragnet measurements as they only provide average values for the whole water column. Thus, we would have to make assumptions about the actual distribution of zooplankton. Further, our regression bound procedure requires a rather dense time grid resolution. The provided biweekly measurements may still be too sparse for making suitable estimations on the error norm bound.

Thus, for now, we could only try our own optimization framework of Section 3.4 in order to get even more evidence that the model is not able to meet the actual data properly.

Chapter 4

A Course Scheduling Problem

4.1 Introduction

The challenge of the Training Management System (TMS) of our corporation partner MINT is to schedule the courses of personal training institutions. Concerning long term planning (up to two years), there may be up to several hundred courses to be held by several hundred instructors for several thousand trainees. High complexity is given by various kinds of constraints, concerning curriculums of courses, qualifications and working rules of human resources, capacities of rooms, etc. We refer to this problem as MINT SP (MINT Scheduling Problem).

Since the foundation of the MINT company in 1998, TMS was extended to support many features that rose from the requests of acquired customers. This lead to a very complex model terminology including some redundant concepts. In joined work with Werner Lehmann from the MINT company, the features of the current scheduling algorithm were worked out and described in terms of an easier structured model [LRS10]. The new model is also intended to allow for an easier integration of future customer demands.

4.1.1 Our Problem

Basic components of the problem model described in [LRS10] are events, locations, resources and assignments. Events represent the atomic parts of courses, e. g., one lesson in aerodynamics, a flight simulation or an examination. Every event has a given duration. Locations are the different places of training centers. Resources may be human resources like instructors, tutors and trainees. Other resources are rooms and technical equipment. Finally, every event has one assignment for every required resource type.

Each assignment specifies a set of qualified resources, including periods of qualification if necessary. Further, resources have limited capacities w.r.t. both the number of simultaneously scheduled events and the allocation of simultaneously scheduled assignments (each assignment will allocate a predefined amount of the capacity of the selected resource).

We use the following notations:

- E: set of all events,
- R: set of all resources,
- L: set of all locations,
- A: set of all assignments,
- d_j, $j \in E$: duration of event j,
- c_r^E, $r \in R$: event capacity of resource r,
- c_r^A, $r \in R$: assignment capacity of resource r,
- R_i^Q, $i \in A$: set of qualified resources for assignment i,
- alloc_i, $i \in A$: resource allocation of assignment i.

The planning horizon is supposed to be divided into sub intervals of equal length (e.g. 15 minutes), all assignment durations and all possible time spaces between two assignments being multiples of this length. Thus, we can find a smallest $N \in \mathbb{N}$ such that every real point in time within the planning horizon may be transformed to a $t \in [N]$. We define

- $T := [N]$: set of normalized points in time.

This set will be used for our time indexed ILP model in Section 4.2. Some resources (especially human resources) are mobile and allowed to travel between different locations. Certain assignments do not allow resources that have to travel longer than a given time limit. We write

- $\tau_{l,l'}$, $l, l' \in L$: required travel time (measured in time grid units) between locations l and l'.
- t_i^{tr}, $i \in A$: maximum allowed travel time for the scheduled resource of assignment i.

A scheduling result assigns unique periods and locations to events and unique resources to assignments, if possible. Some events or assignments may also remain unscheduled.

Based on the above basic components, the complex system of planning rules is formulated by round about 30 hard constraints, which can optionally occur within problem instances. Some constraints also have soft counterparts, formulated as corresponding objectives. We give an overview of the planning rules by listing all constraints with a rough description of their meaning. An instance of the MINT SP provides each constraint by an object which represents all data that is necessary for the exact specification. From the input objects, we derive corresponding parameters that we will use to describe our ILP model. Those parameters are also introduced in the listing below.

▻ **Scheduled event**: Events are either scheduled or unscheduled. Unscheduled Events do neither have a starting time nor a location.

▻ **Event duration**: The actual duration of a scheduled event must accord with its specification.

▻ **Event period allowed start/end**: Either all events or the earliest (latest) event from a given set of events must start within a specified period.
Examples: Specifying the time horizon of a course or allowed times of day for the events of a course.
Parameters:

- $E_{\text{ep}} \subseteq E$: considered events
- $T_{\text{ep}} \subseteq T$: allowed time grid points derived from allowed times of day
- $\text{mode}_{\text{ep}} \in \{\text{startOfFirst}, \text{endOfLast}, \text{startOfAll}\}$: application mode of T_{ep} for E_{ep}
- $T'_{\text{ep}} \subseteq T$: allowed time grid points that correspond to the time horizon for E_{ep}

▻ **Event period minimum distance**: Scheduled events of the event set E_{ep} must not overlap and have to use the earliest possible starting times w.r.t. their chronological scheduling order.
Example: A scheduled course shall not have unnecessary idle time.

▻ **Event period zero distance**: Scheduled events of the event set E_{ep} must take place consecutive and gapless without overlap (more restrictive version of **Event period minimum distance**).

▻ **Event period sequence**: The starting time order of scheduled events of E_{ep} must correspond to a predefined order.
Example: Different subjects of a course build upon one another.
Additional parameter: permutation of the elements of E_{ep}

▻ **Timeframe rule**: This rule defines periods within which certain assignments have to take place. The periods may optionally depend on locations and resources.

▻ **Scheduling group**: Events which belong to the same unit of a course must be scheduled together (either all or none).
Parameters:

- $E_{\text{sg}} \subseteq E$: events of the group
- $\text{AP}_{\text{sg}} \subseteq A \times 2^T$: see subsection 4.2.5

▻ **Assignments running time**: Scheduled assignments of a given assignment set must take place within a bounding period of limited duration.
Example: Courses shall be prevented from being unnecessary long as they require employees to be released from work.
Parameters:

- $A_{\text{ar}} \subseteq A$: considered assignments
- $\text{min}_{\text{ar}}, \text{max}_{\text{ar}}$ limits on the bounding period
- $\text{unit}_{\text{ar}} \in \{\text{minutes}, \text{resourceDays}\}$: measurement unit

▻ **Event alternatives**: Some events are alternatives to each other. For a given system $\mathfrak{E}_{\text{ea}} \subseteq E$ of event sets, at most one event per set must be scheduled.

▻ **Event location**: A rule el of this type defines a set of allowed locations L_{el} for a given event set E_{el}. Moreover, it can be forced that the events have to take place at the same location (by a Boolean flag useSameLocation).
Example: Providing convenient learning conditions.

▻ **Qualified resources**: Resources must be qualified for their assignments.

▻ **Resource event capacity**: Resources $r \in R$ may not be scheduled simultaneously for more than the maximal allowed number of events c_r^E. Typical numbers are 1 for employees and most other resources. Some resources like rooms with working stations allow more than one event.

▻ **Resource assign. capacity**: Resources $r \in R$ may not be scheduled simultaneously for more than their maximum assignment allocation c_r^A (recall that each assignment i has a certain allocation alloc_i).

▻ **Allow non-local resources**: The scheduled resource of certain assignments $i \in A$ must satisfy the given travel time limitation t_i^{tr}.

▻ **Required continuity**: A rule rc of this type defines a set $A_{\mathrm{rc}} \subseteq A$ of assignments that must use the same resource.
Example: All lessons of some subject of a course are desired to be taught by the same trainer.

▻ **Discontinuity**: A rule dc of this type provides two sets $A_{\mathrm{dc},1}, A_{\mathrm{dc},2} \subseteq A$ of assignments. No pair from $A_{\mathrm{dc},1} \times A_{\mathrm{dc},2}$ is allowed to use the same resource.
Example: The trainer of a course is not allowed to be the examiner of the course participants.

▻ **Associated resources**: For certain assignments (*client assignments*) which directly depend on another assignment (*master assignment*) the resource choice may depend on the chosen resource of the master assignment.
Example: Simulators that are associated with theoretical events in certain rooms.
Parameters:

- $a_{\mathrm{ara}}^{\mathrm{master}} \in A$: master assignment
- $A_{\mathrm{ara}}^{\mathrm{client}} \subseteq A$: client assignments
- A set of associated resources $R_{\mathrm{ara},r} \subseteq R$ for each resource $r \in R$ that is qualified for the master assignment

▻ **Resource day**: Working days of human resources (which do not necessary match calender days) must satisfy limits concerning the maximum working time window and the total amount of work.
Parameters:

- $R_{\mathrm{rd}} \subseteq R$: resources that are affected by the rule
- $T_{\mathrm{rd}} \subseteq T$: validity period of the rule in terms of the time grid
- $\mathrm{maxD}_{\mathrm{rd}}$: maximum total amount of work per resource day
- $\mathrm{maxW}_{\mathrm{rd}}$: maximum working time window size per resource day

▻ **Count duty in periods**: There may be limits on the activities of resources within fix periods or rolling periods, respectively. The limits may be universal or restricted to certain locations.
Example: Definition of working rules.
Parameters:

- $R_{\mathrm{cd}} \subseteq R$: considered resources
- $L_{\mathrm{cd}} \subseteq L$: considered locations
- $k_{\mathrm{cd}} \in \mathbb{N}$: total number of periods that have to be checked
- $T_{\mathrm{cd,l,k}} \subseteq T$, $k \in [k_{\mathrm{cd}}]$, $l \in L_{\mathrm{cd}}$: location dependent periods in terms of the time grid
- $\mathrm{min}_{\mathrm{cd}}, \mathrm{max}_{\mathrm{cd}} \in \mathbb{N}$: limits
- $\mathrm{unit}_{\mathrm{cd}} \in \{\mathrm{assignments}, \mathrm{minutes}, \mathrm{resourceDays}\}$: measurement unit

▻ **Count filter in periods**: Similar to **Count duty in periods**, but independent of locations and with the optional restriction to a given set of filter assignments.
Alternatively to the sum of activity durations, this rule may also apply to the duration of consecutive activity.
Example: Maintenance of the subject competence of teachers.
Parameters:

- $R_{\mathrm{cf}} \subseteq R$: considered resources
- $A_{\mathrm{cf}} \subseteq A$: considered filter assignments
- $k_{\mathrm{cf}} \in \mathbb{N}$: total number of periods that have to be checked

- $T_{\mathrm{cf,k}} \subseteq T$, $k \in [k_{\mathrm{cf}}]$: periods in terms of the time grid
- $\mathrm{min}_{\mathrm{cf}} \leqslant \mathrm{max}_{\mathrm{cf}}$: limits
- $\mathrm{unit}_{\mathrm{cf}} \in \{\mathrm{minutes}, \mathrm{resourceDays}\}$: measurement unit

▻ **No resource sharing across locations**: A resource cannot have two locations at the same time.

▻ **Assignment distance**: Bounding periods of the scheduled assignments of two given assignment sets must have a minimum/maximum distance w.r.t. either their start times or the gap in between. Optionally, usage of the first feasible time respecting the gap limit can be required (distance mode minimalGap).
Example: Preparation time for examination. Parameters:

- $A_{\mathrm{ad},1}, A_{\mathrm{ad},2} \subseteq A$: considered assignment sets
- $\mathrm{min}_{\mathrm{ad}} \leqslant \mathrm{max}_{\mathrm{ad}}$: limits
- $\mathrm{unit}_{\mathrm{ad}} \in \{\mathrm{minutes}, \mathrm{resourceDays}\}$: measurement unit
- $\mathrm{mode} \in \{\mathrm{gap}, \mathrm{minimalGap}, \mathrm{startDistance}\}$: distance mode

▻ **Resource day sequence**: Scheduled working days at which certain assignments of a given resource take place must either consecutively use allowed weekdays or comply with minimum and/or maximum gaps which are defined w.r.t. the chronology of the working days.
Example: Definition of appropriate teaching/learning rhythms of a course. Parameters:

- r_{rds}: considered resource
- A_{rds}: considered assignments
- $\delta^{\mathrm{min}}_{\mathrm{rds},k}, \delta^{\mathrm{max}}_{\mathrm{rds},k}$, $k \in [|A_{\mathrm{rds}}| - 1]$: gap limits (optional)

▻ **Resource day assignments**: A certain group of assignments A_{rda} must be scheduled for the same resource and take place at the same working day of this resource.

▻ **Travel time**: If a resource is used at different locations, there must be enough travel time between the corresponding activities.

▻ **Join validity**: Certain events may be joined under corresponding rules which have to be observed.

▻ **Same resource setup time**: Resources may require certain set-up times for certain assignments. If several assignments share a resource, the required set-up time must be the same for all of these assignments.

▻ **Assignment period**: The actual duration of a scheduled assignment must match the duration of the associated event inclusive the possibly required set-up time for the assignment.

▻ **Assignment requires resource and period**: An assignment of a scheduled event must have a selected resource.

4.1.2 Related Work

There is a large amount of literature on the field of scheduling, which concerns exact methods as well as approximation algorithms, heuristics and evolutionary algorithms. We refer to the surveys [Alf04] [ANCK08] [BSW07] [CDL04] [KLPS07] [LL00] [SSW00] [TB01].

The course scheduling problem on hand contains many hard combinatorial subproblems. Variants of algorithms that are dedicated to these subtasks may be utilized for precalculations within an algorithmic framework for the MINT SP. For example, we could reduce the number of variables of the ILP model (Section 4.2) which is the main contribution of this chapter. We give a few example problems along with their relation to the MINT SP.

4.1 Example. Resource Constrained Project Scheduling Problem (RCPSP): Given are n jobs of predefined duration, optional precedence relations between pairs of jobs and m kinds of limited but renewable resources. Each job requires a certain amount of these resources. The objective is to find a schedule with minimum makespan such that precedence relations hold and there are always enough resources available. With respect to the MINT SP the jobs can be parts of courses, e. g., events. Precedence relations concern the order of learning contents as given by **Event period sequence** constraints. Finally, resources are teachers, employees, rooms and equipment.

4.2 Example. Max-k-Cut Problem: Given is a graph $G = (V, \mathcal{E})$ with non-negative edge weights and a number $k \in \mathbb{N}$. The goal is to find a partition of V into k subsets, such that the total weight of inter-partition edges is maximized. Within the MINT SP we can consider V to be a set of employees (teachers) in a region, $\mathcal{E}$ as travel connections and the corresponding weights as travel distances. The parameter k is the number of available training center locations in the region. A maximum weight partition of V is a promising base to assign employees (teachers) to the training centers.

4.3 Example. k-Matching Problem: Here, a hypergraph $H = (V, \mathcal{E})$ is given with non-negative hyperedge weights. A maximum k-matching is a maximum weight subset $M \subseteq \mathcal{E}$ with the property that no vertex of V is contained in more than k hyperedges in M. Concerning the MINT SP one could define V to be a set of teachers and each hyperedge to represent those teachers that are qualified for a certain assignment. The parameter k is a number of events that must not be exceeded by the teachers within the planning horizon (e. g., due to working rules). Then, a maximum k-matching might be a good assignment selection for the schedule.

4.4 Example. k-Set Cover Problem (SCP): Again, a weighted hypergraph $H = (V, \mathcal{E})$ is given. The task is to find a minimum weight subset $S \subseteq \mathcal{E}$, such that at least k vertices are contained in some hyperedge of S. For example, we can define V to be a set of courses that have to be offered and for each available room a hyperedge containing all courses that are possible in that room. Considering the hyperedge weights to be room charges, a minimum k-set cover S is a cheapest room selection such that a (required) selection of least k courses can be assigned to one of the rooms in S.

Some contributions like [FL05][HP11][SMOK06] are about ILP formulations for (parts of) scheduling problems and solution algorithms based upon these.

Due to the limitations of our project, we had to restrict our work to some direction and decided to give a quite comprehensive ILP model of the MINT SP on the one hand and to develop an EA framework on the other hand. The EA framework is the topic of the Diploma Thesis of Torben Rabe [Rab12] and can be understood as a sophisticated extension of the EA approach to the RCPSP described in [Har02]. In the experimental part of this chapter, both the ILP and the EA will be tested for rather small problem instances of the MINT SP. Our intention is to utilize parts of the ILP formulation within heuristics in future.

4.1.3 Complexity of the MINT SP

Similar to the TSPTW considered in Chapter 2, the MINT SP contains the TSP with deadlines problem (see 1.4.1) as a subproblem. Thus, it does not allow for a polynomial algorithm with constant factor approximation (unless $\mathcal{P} = \mathcal{NP}$). For that reason and since the MINT SP contains other $\mathcal{NP}$-hard subproblems (as mentioned above) it is $\mathcal{NP}$-hard itself.

4.2 An ILP Formulation

We now model the MINT SP by a time indexed integer linear program (ILP). For this task, we will deal with decision variables, the number of which will be very large for the whole problem. A time continuous formulation (like that for the TSPTW in Chapter 2) would be much more difficult, here. LP based heuristics, using suitable restrictions of the ILP may help to find approximate solutions of the problem in future.

4.5 Remark. For now, we do not consider assignment setup times nor the option to join certain events. For that reason, a few of the constraints listed above are not modeled by now, namely **Join validity**, **Same resource setup time** and **Assignment period**.

4.6 Remark. we did not implement all the constraints that we have modeled in terms of linear equalities and inequalities, yet. We only implemented constraints that are covered by test instances so far. All implemented constraints is dealt with in this section. The remaining constraints are treated in Section 4.5, namely **Event period minimum distance**, **Assignments running time**, **Event alternatives**, **Event location**, **Allow non-local resources**, **Associated resources**, **No resource sharing across locations** and **Travel time** as well as parts of **Count filter in periods** and **Assignment distance**.

4.2.1 Index sets for the ILP

The most atomic variables (x and y in subsection 4.2.3) decide if

- some event $j \in E$ is active at some point $t \in T$ in time and some location $l \in L$,

- some assignment $i \in A$ is active at some point $t \in T$ in time and some location $l \in L$ while using some resource $r \in R$.

Concerning these variables, there are rules which directly forbid certain combinations in advance (see **Event period allowed start/end**, **Timeframe rule**, **Event location**, **Qualified resources**). Thus, we can precalculate subsets of $E \times T \times L$ and $A \times T \times L \times R$ which only contain allowed combinations and which are much smaller than the sets of all combinations. We introduce

- $J := \{(j,t,l) \in E \times T \times L \,|\, (j,t,l) \text{ allowed}\}$,
- $I := \{(i,t,l,r) \in A \times T \times L \times R \,|\, (i,t,l,r) \text{ allowed}\}$.

If we fix one or more components, only the allowed combinations with other components are important. For example, if we consider a fix assignment with a fix resource, what are the allowed times and locations for them? We formally define accordant index sets as follows. Let $\mathfrak{S}, \mathfrak{S}'$ be disjoint subsystems of $\{A, T, L, R\}$ and $\mathcal{S}, \mathcal{S}'$ be the cartesian products of the sets in $\mathfrak{S}$ and $\mathfrak{S}'$, respectively. For any tuple s' in $\mathcal{S}'$, we define the set $\mathcal{S}_{s'}$ of allowed combinations in $\mathcal{S}$ with respect to s' by

$$\mathcal{S}_{s'} = \left\{ s \in \mathcal{S} \;\middle|\; \begin{array}{l} \text{there is a } \iota \in I \text{ such that each component of } s \text{ and} \\ \text{each component of } s' \text{ appears as a component in } \iota \end{array} \right\}.$$

Similar notations are introduced with $\{E, T, L\}$ and J.

4.7 Example. Let $i \in A$, $l \in L$ and $r \in R$. The set $(T \times L)_{(i,r)}$ contains all allowed combinations of times and locations at which assignment i may be active while using resource r. The set $T_{(i,l,r)}$ contains all allowed times of activity for assignment i at location l with resource r.

4.8 Remark. While our ILP formulations is based on the reduced index sets, our implementation does not yet precalculate such sets but uses the full index sets E, R, L, A and T. On the other hand, our implementation does not deal with multiple locations. Thus, it uses location-free indices for all ILP variables that are introduced below.

4.2.2 Basic Variables

A scheduling result mainly depends on five types of basic decisions. For each event we have to decide, if it gets scheduled or not. The same holds for every assignment. We further have to determine a unique period and a unique location for every scheduled event and a unique resource for every scheduled assignment. We introduce indicator variables for this tasks:

- $x_j^X \in \{0,1\}$, $j \in E$
  x_j^X is 1, if and only if event j is scheduled.
- $x_{j,t}^{\text{ES}} \in \{0,1\}$, $j \in E, t \in T_{(j)}$
  $x_{j,t}^{\text{ES}}$ is 1, if and only if event j starts at time t.
- $x_{j,l}^L \in \{0,1\}$, $j \in E, l \in L_{(j)}$
  $x_{j,l}^L$ is 1, if and only if event j takes place at location l.
- $y_i^X \in \{0,1\}$, $i \in A$
  y_i^X is 1, if and only if assignment i is scheduled.
- $y_{i,r}^R \in \{0,1\}$, $i \in A, r \in R_{(i)}$
  $y_{i,r}^R$ is 1, if and only if resource r is assigned to i.

4.2.3 Auxiliary Variables

From the basic variables, we derive a couple of auxiliary variables which make it easier to express the large number of objectives and constraints of the MINT SP in terms of linear equalities and inequalities. We introduce

- $\text{ES}_j \in T$, $j \in E$
  ES_j is the start time of event j.
- $x_{j,t,l} \in \{0,1\}$, $j \in E, t \in T_{(j)}, l \in L_{(j,t)}$
  $x_{j,t,l}$ is 1, if and only if event j is active at time t and location l.
- $\text{AS}_i \in T$, $i \in A$
  AS_i is the start time of assignment i.
- $y_{i,t}^{\text{AS}} \in \{0,1\}$, $i \in A, t \in T_{(i)}$
  $y_{i,t}^{\text{AS}}$ is 1, if and only if assignment i starts at time t.
- $y_{i,t}^A \in \{0,1\}$, $i \in A, t \in T_{(i)}$
  $y_{i,t}^A$ is 1, if and only if assignment i is active at time t.
- $y_{i,l}^L \in \{0,1\}$, $i \in A, l \in L_{(i)}$
  $y_{i,l}^L$ is 1, if and only if assignment i takes place at location l.

- $y^{\text{ASLR}}_{i,t,l,r} \in \{0,1\}$, $i \in A, t \in T_{(i)}, l \in L_{(i,t)}, r \in R_{(i,t,l)}$
  $y^{\text{ASLR}}_{i,t,l,r}$ is 1, if and only if assignment i starts at time t and location l and uses resource r.

- $y_{i,t,l,r} \in \{0,1\}$, $i \in A, t \in T_{(i)}, l \in L_{(i,t)}, r \in R_{(i,t,l)}$
  $y_{i,t,l,r}$ is 1, if and only if assignment i is active at time t and location l while using resource r.

- $z_{r,t} \in \{0,1\}$, $r \in R, t \in T_{(r)}$
  $z_{r,t}$ is 1, if and only if resource r is active at time t.

Due to working rules, many constraints concern the working days of human resources (e. g., two days off within every time period of seven days). Working days are generalized to resource days, as there may occur similar rules for non human resources in future. Between two consecutive resource days there must be an activity-free period of regeneration which has a specified minimum length called turntime. For a given solution and a given resource we need to identify the resource day of every assignment. This is done by identifying working blocks of the resource. A working block of a resource is a time interval which is maximal w.r.t. the property that the resource is active at its bounds and that it does not contain a subinterval of turntime length within which the resource is inactive. Every start of a working block marks a day break and no further day break is contained in a working block. The number of additional day breaks in the gap between two consecutive working blocks is defined by the gap size. For an $r \in R$ with turntime n^{turn}_r, both the number of working blocks within T and the number of resource days within T are limited by $W_r := \lceil N/n^{\text{turn}}_r \rceil$. We introduce the following auxiliary variables to express working rules:

- $z^{\text{WS}}_{r,j,t} \in \{0,1\}$, $j \in [W_r]$, $r \in R$, $t \in T_{(r)} \cup \{N+1, \cdots, N+W_r\}$
  $z^{\text{WS}}_{r,j,t}$ is 1, if and only if j-th working block of resource r starts at time t.

- $z^{\text{WE}}_{r,j,t} \in \{0,1\}$, $j \in [W_r]$, $r \in R$, $t \in T_{(r)} \cup \{N+1, \cdots, N+W_r\}$
  $z^{\text{WE}}_{r,j,t}$ is 1, if and only if j-th working block of resource r ends at time t.

- $\text{WS}_{r,j} \in T_{(r)} \cup \{N+1, \cdots, N+W_r\}$, $r \in R$, $j \in [W_r]$
  $\text{WS}_{r,j}$ is the start time of the j-th working block of resource r.

- $\text{WE}_{r,j} \in T_{(r)} \cup \{N+1, \cdots, N+W_r\}$, $r \in R$, $j \in [W_r]$
  $\text{WE}_{r,j}$ is the finish time of the j-th working block of resource r.

- $z^{\text{WA}}_{r,j,t} \in \{0,1\}$, $j \in [W_r]$, $r \in R$, $t \in T_{(r)} \cup \{N+1, \cdots, N+W_r\}$
  $z^{\text{WA}}_{r,j,t}$ is 1, if and only if resource r is active within its j-th working block at time t.
- $\text{GD}_{r,j} \in [W_r]$, $r \in R$, $j \in [W_r - 1]$
  $\text{GD}_{r,j}$ is for resource r the number of resource days in the gap between working block j and working block $j+1$.
- $\text{WD}_{r,j} \in [W_r]$, $r \in R$, $j \in [W_r]$
  $\text{WD}_{r,j}$ is the resource day which contains the j-th working block of resource r.
- $y^{\text{before}}_{r,i,j} \in \{0,1\}$, $r \in R$, $i \in A$, $j \in [W_r]$
  $y^{\text{before}}_{r,i,j}$ is 1, if and only if assignment i starts before j-th working block of resource r.
- $\overline{\text{ARD}}_{r,i} \in [W_r]$, $r \in R$, $i \in A$
  $\overline{\text{ARD}}_{r,i}$ is the resource day of r at which assignment i takes place.
- $y^{\overline{\text{ARD}}}_{i,r,d} \in \{0,1\}$, $i \in A$, $r \in R_{(i)}$, $d \in [W_r]$
  $y^{\overline{\text{ARD}}}_{i,r,d}$ is 1, if and only if assignment i takes place at the d-th resource day of r.
- $y^{\text{ARD}}_{i,r,d} \in \{0,1\}$, $i \in A$, $r \in R_{(i)}$, $d \in [W_r]$
  $y^{\text{ARD}}_{i,r,d}$ is 1, if and only if assignment i is assigned to resource r and takes place at its d-th resource day.
- $y^{\text{ADLR}}_{i,d,l,r} \in \{0,1\}$, $i \in A$, $r \in R_{(i)}$, $d \in [W_r]$, $l \in L_{(i,r)}$
  $y^{\text{ADLR}}_{i,d,l,r}$ is 1, if and only if assignment i uses resource r and takes place at location l and the d-th resource day of r.
- $y^{\text{AD}}_{i,d} \in \{0,1\}$, $i \in A$, $d \in [W_r]$
  $y^{\text{AD}}_{i,d}$ is 1, if and only if i is assigned to a certain resource r_i and takes place at the d-th resource day of r_i.
- $\text{AD}_i \in [W_{\text{ad}}]$, $i \in A$ and $W_{\text{ad}} = W_r$ where $r \in R_{(i)}$ is any suitable resource for i (it is supposed that $W_r = W_{r'}$ for every other suitable resources $r' \in R$)
  If i uses resource r_i, AD_i is the resource day of i with respect to r_i.

Let rds be a **Resource day sequence** constraint and let r_{rds} and A_{rds} be the associated resource and the considered set of assignments, respectively.

We introduce the auxiliary variables

- $\mathrm{SD}_{\mathrm{rds},k} \in [W_{\mathrm{rds}} + 1]$, $k \in [|A_{\mathrm{rds}}|]$
  $\mathrm{SD}_{\mathrm{rds},k}$ is the resource day of the k-th working block w.r.t. r_{rds} and A_{rds} where $W_{\mathrm{rds}} + 1$ represents a dummy working block
- $y^{\mathrm{SD}}_{\mathrm{rds},k,d} \in \{0,1\}$, $k \in [|A_{\mathrm{rds}}|]$, $d \in [W_{\mathrm{rds}} + 1]$
  $y^{\mathrm{SD}}_{\mathrm{rds},k,d}$ is 1, if and only if $\mathrm{SD}_{\mathrm{rds},k}$ is d

In order to verify if assignments consecutively use allowed weekdays, we need to indicate times which lie between the start of the first scheduled assignment in A_{rds} and the end of the last scheduled assignment in A_{rds}. We introduce the variables

- $y^{\mathrm{RDS1}}_{\mathrm{rds},t} \in \{0,1\}$, $t \in T$
  $y^{\mathrm{RDS1}}_{\mathrm{rds},t}$ is 1, if and only if r_{rds} is active in $[0,t]$ for some assignment in A_{rds}
- $y^{\mathrm{RDS2}}_{\mathrm{rds},t} \in \{0,1\}$, $t \in T$
  $y^{\mathrm{RDS2}}_{\mathrm{rds},t}$ is 1, if and only if r_{rds} is active in $[t,T]$ for some assignment in A_{rds}
- $y^{\mathrm{RDS3}}_{\mathrm{rds},t} \in \{0,1\}$, $t \in T$
  $y^{\mathrm{RDS3}}_{\mathrm{rds},t}$ is 1, if and only if both $y^{\mathrm{RDS1}}_{\mathrm{rds},t}$ and $y^{\mathrm{RDS2}}_{\mathrm{rds},t}$ are 1.

Let cf be a **Count filter in periods** rule. Let R_{cf}, A_{cf} be the corresponding sets of resources and assignments, respectively. All resources in R_{cf} are supposed to have the same turntime, i.e the same maximum number W_{cf} of resource days. The constraint cf further defines if either active times or inactive times must be considered. If cf concerns the total amount of activity (inactivity), for every $r \in R_{\mathrm{cf}}$ we need to indicate times which lie between the resource days of the first scheduled assignment in A_{cf} and the last scheduled assignment in A_{cf}. Thus, we introduce similar variables as for the **Resource day sequence** constraint above:

- $y^{\mathrm{CF1}}_{\mathrm{cf},r,d} \in \{0,1\}$, $r \in R_{\mathrm{cf}}$, $d \in [W_{\mathrm{cf}}]$
  $y^{\mathrm{CF1}}_{\mathrm{cf},r,d}$ is 1, if and only if r is active at a resource day in $[0,d]$ for some assignment in A_{cf}
- $y^{\mathrm{CF2}}_{\mathrm{cf},r,d} \in \{0,1\}$, $r \in R_{\mathrm{cf}}$, $d \in [W_{\mathrm{cf}}]$
  $y^{\mathrm{CF2}}_{\mathrm{cf},r,d}$ is 1, if and only if r is active at a resource day in $[d, W_{\mathrm{cf}}]$ for some assignment in A_{cf}

- $y^{\text{CF3}}_{\text{cf},r,d} \in \{0,1\}$, $r \in R_{\text{cf}}$, $d \in [W_{\text{cf}}]$
  $y^{\text{CF3}}_{\text{cf},r,d}$ is 1, if and only if both $y^{\text{CF1}}_{\text{cf},r,d}$ and $y^{\text{CF2}}_{\text{cf},r,d}$ are 1.

4.2.4 The Relationships between Variables

Relationships between basic variables and auxiliary variables are given by the following constraints. We first state that every scheduled event $j \in E$ has a unique start time ES_j and that an unscheduled event has no start time (4.1)(4.2). The start time of a scheduled event $j \in E$ has the property that j is active at times $t \in \{\text{ES}_j, \cdots, \text{ES}_j + d_j - 1\}$ (4.3) but at no other time and unscheduled events can't be active at any time (4.4). For a given event $j \in E$ and a given time $t \in T_{(j)}$ we use the set $T^t_j := T_{(j)} \cap \{t, \cdots, t + d_j - 1\}$.

$$x^X_j = \sum_{t \in T_{(j)}} x^{\text{ES}}_{j,t} \quad \text{for all } j \in E \tag{4.1}$$

$$\text{ES}_j = \sum_{t \in T_{(j)}} t \cdot x^{\text{ES}}_{j,t} \quad \text{for all } j \in E \tag{4.2}$$

$$\sum_{s \in T^t_j} \sum_{l \in L_{(j,t)}} x_{j,s,l} \geqslant |T^t_j| \cdot x^{\text{ES}}_{j,t} \quad \text{for all } j \in E, t \in T_{(j)} \tag{4.3}$$

$$\sum_{t \in T_{(j)}} \sum_{l \in L_{(j,t)}} x_{j,t,l} = d_j \cdot x^X_j \quad \text{for all } j \in E \tag{4.4}$$

A scheduled event has exactly one location and an unscheduled event has no location (4.5). An event has location l, if it is active there at some point in time (4.6).

$$\sum_{l \in L_{(j)}} x^L_{j,l} = x^X_j \quad \text{for all } j \in E \tag{4.5}$$

$$d_j \cdot x^L_{j,l} \geqslant \sum_{t \in T_{(j,l)}} x_{j,t,l} \quad \text{for all } j \in E, l \in L_{(j)} \tag{4.6}$$

An assignment can only be scheduled, if the accordant event is scheduled (4.7) and the start time of a scheduled assignment is (for now) the start time of its event (4.8)(4.9). The location of a scheduled assignment is the location of its event (4.10).

$$y^X_i \leqslant x^X_{j(i)} \quad \text{for all } i \in A \tag{4.7}$$

$$y^{\mathrm{AS}}_{i,t} \leqslant x^{\mathrm{ES}}_{j(i),t} \quad \text{for all } i \in A, t \in T_{(j(i))} \tag{4.8}$$

$$y^{\mathrm{AS}}_{i,t} \geqslant x^{\mathrm{ES}}_{j(i),t} + y^X_i - 1 \quad \text{for all } i \in A, t \in T_{(j(i))} \tag{4.9}$$

$$y^L_{i,l} = x^L_{j(i),l} \quad \text{for all } i \in A, l \in L_{(i)} \tag{4.10}$$

Similar to the relationships (4.1)-(4.4), we have relationships between the start time indicators of assignments with their schedule state indicators (4.11), start times (4.12) and indicators for times of activity (4.13)(4.14). We use the duration $\overline{d}_i$ of an assignment $i \in A$, which is supposed to be the duration of its event $j(i)$ as we do not consider setup times, yet. For $i \in A$ and $t \in T_{(i)}$, we use the set $T^t_i := T_{(i)} \cap \{t, \cdots, t + \overline{d}_i - 1\}$.

$$y^X_i = \sum_{t \in T_{(i)}} y^{\mathrm{AS}}_{i,t} \quad \text{for all } i \in A \tag{4.11}$$

$$\mathrm{AS}_i = \sum_{t \in T_{(i)}} t \cdot y^{\mathrm{AS}}_{i,t} \quad \text{for all } i \in A \tag{4.12}$$

$$\sum_{s \in T^t_i} y^A_{i,s} \geqslant |T^t_i| \cdot y^{\mathrm{AS}}_{i,t} \quad \text{for all } i \in A, t \in T_{(i)} \tag{4.13}$$

$$\sum_{t \in T_{(i)}} y^A_{i,t} = \overline{d}_i \cdot y^X_i \quad \text{for all } i \in A \tag{4.14}$$

An assignment is active at time t, if and only if it is active at time t and some location l while using some resource r.

$$y^A_{i,t} = \sum_{l \in L_{(i,t)}} \sum_{r \in R_{(i,t,l)}} y_{i,t,l,r} \quad \text{for all } i \in A, t \in T_{(i)} \tag{4.15}$$

Every scheduled assignment has a unique resource (4.16). An assignment has a certain resource r, if it is active at some time and some location while using r (4.17).

$$\sum_{r \in R} y^R_{i,r} \geqslant y^X_i \quad \text{for all } i \in A \tag{4.16}$$

$$\overline{d}_i \cdot y^R_{i,r} \geqslant \sum_{t \in T_{(i,r)}} \sum_{l \in L_{(i,t,r)}} y_{i,t,l,r} \quad \text{for all } i \in A, r \in R_{(i)} \tag{4.17}$$

The indicators $y_{i,t,l,r}$ and $y^{\mathrm{ASLR}}_{i,t,l,r}$ of combined assignment information are derived from the corresponding single information indicators by (4.18)(4.19), where the minimum notations actually mean 3 inequalities.

$$y^{A}_{i,t} + y^{L}_{i,l} + y^{R}_{i,r} - 2 \leqslant y_{i,t,l,r} \leqslant \min\{y^{A}_{i,t}, y^{L}_{i,l}, y^{R}_{i,r}\}$$
$$\text{for all } i \in A, t \in T_(i), l \in L_{(i,t)}, r \in R_{(i,t,l)} \tag{4.18}$$

$$y^{\mathrm{AS}}_{i,t} + y^{L}_{i,l} + y^{R}_{i,r} - 2 \leqslant y^{ASLR}_{i,t,l,r} \leqslant \min\{y^{\mathrm{AS}}_{i,t}, y^{L}_{i,l}, y^{R}_{i,r}\}$$
$$\text{for all } i \in A, t \in T_(i), l \in L_{(i,t)}, r \in R_{(i,t,l)} \tag{4.19}$$

A resource is active at time t if and only if there is an assignment i and a location l such that i is active at time t and location l while using resource r (4.20)(4.21).

$$z_{r,t} \geqslant \frac{1}{C^{E}_{r}} \cdot \sum_{i \in A_{(t,r)}} \sum_{l \in L_{(i,t,r)}} y_{i,t,l,r} \quad \text{for all } r \in R, t \in T_{(r)} \tag{4.20}$$

$$z_{r,t} \leqslant \sum_{i \in A_{(t,r)}} \sum_{l \in L_{(i,t,r)}} y_{i,t,l,r} \quad \text{for all } r \in R, t \in T_{(r)} \tag{4.21}$$

Every working block of every resource has a unique start time and a unique finish time

$$\sum_{t \in T_{(r)}} x^{\mathrm{WS}}_{r,j,t} + \sum_{t=N+1}^{N+W_r} x^{\mathrm{WS}}_{r,j,t} = 1 \quad \text{for all } r \in R, j \in [W_r] \tag{4.22}$$

$$\sum_{t \in T_{(r)}} x^{\mathrm{WE}}_{r,j,t} + \sum_{t=N+1}^{N+W_r} x^{\mathrm{WE}}_{r,j,t} = 1 \quad \text{for all } r \in R, j \in [W_r] \tag{4.23}$$

which are determined as

$$\mathrm{WS}_{r,j} = \sum_{t \in T_{(r)}} t \cdot x^{\mathrm{WS}}_{r,j,t} + \sum_{N+1}^{N+W_r} t \cdot x^{\mathrm{WS}}_{r,j,t} \quad \text{for all } r \in R, j \in [W_r], \tag{4.24}$$

$$\mathrm{WE}_{r,j} = \sum_{t \in T_{(r)}} t \cdot x^{\mathrm{WE}}_{r,j,t} + \sum_{N+1}^{N+W_r} t \cdot x^{\mathrm{WE}}_{r,j,t} \quad \text{for all } r \in R, j \in [W_r]. \tag{4.25}$$

The finish time of one working block is not before its start time but before the start time of the next working block (in particular, the start times of the working blocks of a resource are in ascending order):

$$\mathrm{WS}_{r,j} \leqslant \mathrm{WE}_{r,j} \quad \text{for all } r \in R, j \in [W_r], \tag{4.26}$$

$$\mathrm{WE}_{r,j} < \mathrm{WS}_{r,j+1} \quad \text{for all } r \in R, j \in [W_r - 1]. \tag{4.27}$$

For every $r \in R$ and every $t \in T_{(r)}$ there starts a working block of r at t, if and only if r is assigned at t but unassigned within the preceding time interval of turntime length. The first implication of this equivalence is expressed by (4.28) and (4.29). The other implication is given by (4.30). We use the set $T_r^{t-} := T_{(r)} \cap \{t - n_r^{\mathrm{turn}}, \cdots, t-1\}$ for $r \in R$ and $t \in T_{(r)}$.

$$\sum_{j \in [W_r]} x_{r,j,t}^{\mathrm{WS}} \leqslant z_{r,t} \quad \text{for all } r \in R, t \in T_{(r)}, \tag{4.28}$$

$$\sum_{s \in T_r^{t-}} z_{r,s} \leqslant |T_r^{t-}| \cdot (1 - \sum_{j \in [W_r]} x_{r,j,t}^{\mathrm{WS}}) \quad \text{for all } r \in R, t \in T_{(r)}, \tag{4.29}$$

$$z_{r,t} - \sum_{s \in T_r^{t-}} z_{r,s} \leqslant \sum_{j \in [W_r]} x_{r,j,t}^{\mathrm{WS}} \quad \text{for all } r \in R, t \in T_{(r)}. \tag{4.30}$$

Similarly, for every $r \in R$ and every $t \in T_{(r)}$ there ends a working block of r at t, if and only if r is assigned at t but unassigned within the following time interval of turntime length. Here, we use the set $T_r^{t+} := T_{(r)} \cap \{t+1, \cdots, t + n_r^{\mathrm{turn}}\}$ for $r \in R$ and $t \in T_{(r)}$.

$$\sum_{j \in [W_r]} x_{r,j,t}^{\mathrm{WE}} \leqslant z_{r,t} \quad \text{for all } r \in R, t \in T_{(r)}, \tag{4.31}$$

$$\sum_{s \in T_r^{t+}} z_{r,s} \leqslant |T_r^{t+}| \cdot (1 - \sum_{j \in [W_r]} x_{r,j,t}^{\mathrm{WE}}) \quad \text{for all } r \in R, t \in T_{(r)}, \tag{4.32}$$

$$z_{r,t} - \sum_{s \in T_r^{t+}} z_{r,s} \leqslant \sum_{j \in [W_r]} x_{r,j,t}^{\mathrm{WE}} \quad \text{for all } r \in R, t \in T_{(r)}. \tag{4.33}$$

4.9 Remark. Working blocks that start/end in T are actual working blocks. Dummy working blocks start/end in $\{N+1, \cdots, N+W_r\}$.

4.10 Remark. Gaps in $T_{(r)}$, $r \in R$, are supposed to be longer than n_r^{turn}.

A variable $z_{r,j,t}^{\mathrm{WA}}$ must be 1, if and only if $z_{r,t}$ is 1 and $\mathrm{WS}_j \leqslant t \leqslant \mathrm{WE}_j$. The implications of this equivalence are given by (4.34) and (4.35), respectively, where the minimum notation actually means 3 inequalities.

$$z_{r,j,t}^{\mathrm{WA}} \leqslant \min\{z_{r,t}, \sum_{s \leqslant t} z_{r,j,s}^{\mathrm{WS}}, \sum_{s \geqslant t} z_{r,j,s}^{\mathrm{WE}}\} \quad \text{for all } r \in R, t \in T_{(r)}, j \in [W_r] \tag{4.34}$$

$$z^{\text{WA}}_{r,j,t} \geqslant \sum_{s \leqslant t} z^{\text{WS}}_{r,j,s} + \sum_{s \geqslant t} z^{\text{WE}}_{r,j,s} + z_{r,t} - 2 \quad \text{for all } r \in R, t \in T_{(r)}, j \in [W_r] \tag{4.35}$$

For every resource $r \in R$ and every working block $j \in [W_r - 1]$, the number of resource days in the gap between working block j and working block $j+1$ is determined by

$$\text{GD}_{r,j} \leqslant \frac{\text{WS}_{r,j+1} - \text{WE}_{r,j}}{n^{\text{day}}} + \frac{1}{4} - \frac{1}{n^{\text{day}}}, \tag{4.36}$$

$$\text{GD}_{r,j} \geqslant \frac{\text{WS}_{r,j+1} - \text{WE}_{r,j}}{n^{\text{day}}} - \frac{3}{4}, \tag{4.37}$$

where n^{day} is the number time grid units for 24 hours. This means that a gap counts no resource day if its duration is less or equal to 18 hours, one resource day if its duration is greater then 18 hours but less or equal to 42 $(18 + 24)$ hours, and so forth. The resource day of a working block is equal to the number of preceding working blocks plus the sum of resource days in the gaps in between:

$$\text{WD}_{r,j} = j - 1 + \sum_{k \in [j-1]} \text{GD}_{r,k} \quad \text{for all } r \in R. \tag{4.38}$$

The following inequalities define the variables that indicate, if an assignment i starts before the j-th working block of a resource r:

$$y^{\text{before}}_{r,i,j} \leqslant \frac{\text{WS}_{r,j} - \text{AS}_i - 1}{T} + 1 \quad \text{for all } i \in A, r \in R_{(i)}, j \in [W_r], \tag{4.39}$$

$$y^{\text{before}}_{r,i,j} \geqslant \frac{\text{WS}_{r,j} - \text{AS}_i - 1}{T} \quad \text{for all } i \in A, r \in R_{(i)}, j \in [W_r]. \tag{4.40}$$

Actually, the above indicator variables are undefined, if $\text{WS}_{r,j} = \text{AS}_i + 1$, but this can't happen if resource r is assigned to i. Now, the resource day of an $r \in R$ at which an assignment $i \in A$ takes place can be determined by

$$\overline{\text{ARD}}_{r,i} \leqslant \text{WD}_{r,j} + W_r(1 - y^{\text{before}}_{r,i,j+1}) \quad \text{for all } i \in A, r \in R_{(i)}, j \in [W_r - 1], \tag{4.41}$$

$$\overline{\text{ARD}}_{r,i} \geqslant \text{WD}_{r,j} - W_r y^{\text{before}}_{r,i,j} \quad \text{for all } i \in A, r \in R_{(i)}, j \in [W_r]. \tag{4.42}$$

The relationships between those variables and the corresponding indicator variables are given by (4.43)(4.44) and the variables which indicate if an assignment i uses a resource r at a certain resource day d of r are derived by (4.45)(4.46)(4.47). Indicators if assignments take place at certain resource days of their own resource can be derived by (4.48)(4.49).

$$\sum_{d \in [W_r]} y^{\overline{\mathrm{ARD}}}_{r,i,d} \leqslant y^X_i \quad \text{for all } i \in A, r \in R_{(i)} \tag{4.43}$$

$$\sum_{d \in [W_r]} d \cdot y^{\overline{\mathrm{ARD}}}_{r,i,d} = \overline{\mathrm{ARD}}_{r,i} \quad \text{for all } i \in A, r \in R_{(i)} \tag{4.44}$$

$$y^{\mathrm{ARD}}_{i,r,d} \leqslant y^{\overline{\mathrm{ARD}}}_{i,r,d} \quad \text{for all } i \in A, r \in R_{(i)}, d \in W_r \tag{4.45}$$

$$y^{\mathrm{ARD}}_{i,r,d} \leqslant y^R_{i,r} \quad \text{for all } i \in A, r \in R_{(i)}, d \in W_r \tag{4.46}$$

$$y^{\mathrm{ARD}}_{i,r,d} \geqslant y^{\overline{\mathrm{ARD}}}_{i,r,d} + y^R_{i,r} - 1 \quad \text{for all } i \in A, r \in R_{(i)}, d \in W_r \tag{4.47}$$

$$\sum_{d \in [W_{\mathrm{ad}}]} y^{\mathrm{AD}}_{i,d} = y^X_i \quad \text{for all } i \in A \tag{4.48}$$

$$y^{\mathrm{AD}}_{i,d} = \sum_{r \in R_{(i)}} y^{\mathrm{ARD}}_{i,r,d} \quad \text{for all } i \in A, d \in W_{\mathrm{ad}} \tag{4.49}$$

The y^{ADLR} variables are derived from the y^{ARD} variables and the y^L variables.

$$\frac{y^{\mathrm{ARD}}_{i,r,d} + y^L_{i,l}}{2} \leqslant y^{\mathrm{ADLR}}_{i,d,l,r} \leqslant \min\{y^{\mathrm{ARD}}_{i,r,d}, y^L_{i,l}\}$$
$$\text{for all } i \in A, r \in R_{(i)}, d \in [W_r], l \in L_{(i,r)} \tag{4.50}$$

Finally, the resource days of scheduled assignments w.r.t. their resource are

$$\mathrm{AD}_i = \sum_{d \in [W_{\mathrm{ad}}]} d \cdot y^{\mathrm{AD}}_{i,d} \quad \text{for all } i \in A \tag{4.51}$$

Variables that indicate if the k-th active resource day of a **Resource day sequence** constraint rds with resource r_{rds} and assignments A_{rds} is day d are given by

$$y^{\mathrm{SD}}_{\mathrm{rds},k,d} \leqslant \sum_{i \in A_{\mathrm{rds}}} y^{\mathrm{AD}}_{i,d} \quad \text{for all } d \in [W_{\mathrm{sd}} + 1], k \in [|A_{\mathrm{rds}}|] \tag{4.52}$$

$$y^{\mathrm{SD}}_{\mathrm{rds},k,d} \geqslant \frac{1}{|A_{\mathrm{rds}}|} \sum_{i \in A_{\mathrm{rds}}} y^{\mathrm{AD}}_{i,d} \quad \text{for all } d \in [W_{\mathrm{sd}} + 1], k \in [|A_{\mathrm{rds}}|] \tag{4.53}$$

$$\sum_{d \in [W_{\mathrm{sd}}+1]} y^{\mathrm{SD}}_{\mathrm{rds},k,d} = 1 \quad \text{for all } k \in [|A_{\mathrm{rds}}|] \tag{4.54}$$

and the corresponding k-th active resource day is

$$\mathrm{SD}_{\mathrm{rds},k} = \sum_{d \in [W_{\mathrm{sd}}+1]} d \cdot y^{\mathrm{SD}}_{\mathrm{rds},k,d} \quad \text{for all } k \in [|A_{\mathrm{rds}}|]. \tag{4.55}$$

Variables that indicate times between the start of the first scheduled assignment in A_{rds} and the end of the last scheduled assignment in A_{rds} are derived as follows:

$$y^{\mathrm{RDS1}}_{\mathrm{rds},t} \leqslant \sum_{i \in A_{\mathrm{rds}}} \sum_{s=1}^{t} y_{i,r_{\mathrm{rds}},s} \quad \text{for all } t \in T \tag{4.56}$$

$$y^{\mathrm{RDS1}}_{\mathrm{rds},t} \geqslant \frac{1}{d_{\mathrm{rds}}} \sum_{i \in A_{\mathrm{rds}}} \sum_{s=1}^{t} y_{i,r_{\mathrm{rds}},s} \quad \text{for all } t \in T \tag{4.57}$$

$$y^{\mathrm{RDS2}}_{\mathrm{rds},t} \leqslant \sum_{i \in A_{\mathrm{rds}}} \sum_{s=t}^{T} y_{i,r_{\mathrm{rds}},s} \quad \text{for all } t \in T \tag{4.58}$$

$$y^{\mathrm{RDS2}}_{\mathrm{rds},t} \geqslant \frac{1}{d_{\mathrm{rds}}} \sum_{i \in A_{\mathrm{rds}}} \sum_{s=t}^{T} y_{i,r_{\mathrm{rds}},s} \quad \text{for all } t \in T \tag{4.59}$$

$$y^{\mathrm{RDS3}}_{\mathrm{rds},t} \leqslant y^{\mathrm{RDS1}}_{\mathrm{rds},t} \quad \text{for all } t \in T \tag{4.60}$$

$$y^{\mathrm{RDS3}}_{\mathrm{rds},t} \leqslant y^{\mathrm{RDS2}}_{\mathrm{rds},t} \quad \text{for all } t \in T \tag{4.61}$$

$$y^{\mathrm{RDS3}}_{\mathrm{rds},t} \geqslant y^{\mathrm{RDS1}}_{\mathrm{rds},t} + y^{\mathrm{RDS2}}_{\mathrm{rds},t} - 1 \quad \text{for all } t \in T \tag{4.62}$$

with $d_{\mathrm{rds}} = \sum_{i \in A_{\mathrm{rds}}} \overline{d}_i$.

Similarly, for every **Count filter in periods** constraint cf with resources R_{cf} and assignments A_{cf} the variables that indicate times between the start of the first scheduled assignment in A_{cf} and the end of the last scheduled assignment in A_{cf} are derived by

$$y^{\mathrm{CF1}}_{\mathrm{cf},r,d} \leqslant \sum_{i \in A_{\mathrm{cf}}} \sum_{s=1}^{d} y_{i,r,s} \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.63}$$

$$y^{\mathrm{CF1}}_{\mathrm{cf},r,d} \geqslant \frac{1}{|A_{\mathrm{cf}}|} \sum_{i \in A_{\mathrm{cf}}} \sum_{s=1}^{d} y_{i,r,s} \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.64}$$

$$y^{\mathrm{CF2}}_{\mathrm{cf},r,d} \leqslant \sum_{i \in A_{\mathrm{cf}}} \sum_{s=d}^{W_{\mathrm{cf}}} y_{i,r,s} \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.65}$$

$$y^{\mathrm{CF2}}_{\mathrm{cf},r,d} \geqslant \frac{1}{|A_{\mathrm{cf}}|} \sum_{i \in A_{\mathrm{cf}}} \sum_{s=d}^{W_{\mathrm{cf}}} y_{i,r,s} \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.66}$$

$$y^{\mathrm{CF3}}_{\mathrm{cf},r,d} \leqslant y^{\mathrm{CF1}}_{\mathrm{cf},r,d} \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.67}$$

$$y^{\mathrm{CF3}}_{\mathrm{cf},r,d} \leqslant y^{\mathrm{CF2}}_{\mathrm{cf},r,d} \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.68}$$

$$y^{\mathrm{CF3}}_{\mathrm{cf},r,d} \geqslant y^{\mathrm{CF1}}_{\mathrm{cf},r,d} + y^{\mathrm{CF2}}_{\mathrm{cf},r,d} - 1 \quad \text{for all } r \in R_{\mathrm{cf}}, d \in [W_{\mathrm{cf}}] \tag{4.69}$$

4.2.5 Objectives and Constraints of the MINT SP

We are now ready to express many objectives and constraints of the MINT SP.

Main objective is to maximize the profit of the training center, which is the difference of all revenues (fees for the assignments) and expenses (resource costs). Equivalently, we minimize the negated profit:

$$\min \sum_{r \in R} \sum_{t \in T_{(r)}} \rho_{r,t} \cdot z_{r,t} - \sum_{i \in A} x_i^X \alpha_i, \tag{4.70}$$

where the different types of expenses and revenues used in (4.70) are given by instance:

- time dependent resource costs per time unit $\rho_{r,t}$, $r \in R$, $t \in T$
- fees α_i for assignments $i \in A$

Scheduled event: The fact that events are either scheduled or unscheduled and that an unscheduled event does neither have a period nor a location is stated by (4.1) and (4.5).

Event duration: The period of a scheduled event must have the requested duration. This condition is expressed by (4.3) and (4.4).

Event period allowed start/end: For every constraint ep of this type let $E_{\mathrm{ep}} \subseteq E$ be the associated set of events, $T_{\mathrm{ep}} \subseteq T$ the set of all time grid points that correspond to allowed start periods (given in terms of calender date intervals) and $T'_{\mathrm{ep}} \subseteq T$ the set of all time grid points that correspond to allowed start times (given in terms of week times).

4.11 Remark. Deviating from [LRS10], we restrict to the case that the given events have a common predefined location, yet.

Now, if $T_{\text{ep}} \neq T$, we must enforce that the first scheduled event in E_{ep} starts at a time $t \in T_{\text{ep}}$. This is done by (4.71a), because its negation means that there exists a $t \in T$ such that no event starts in $T_{\text{ep}} \cap \{1, \cdots, t\}$ but some event starts in $(T \backslash T_{\text{ep}}) \cap \{1, \cdots, t\}$.

$$\sum_{\substack{s \in T_{\text{ep}} \\ s \leqslant t}} \sum_{\substack{j \in E_{\text{ep}} \\ j \in E_{(s)}}} |E_{\text{ep}} \cap E_{(s)}| \cdot x^{\text{ES}}_{j,s} - \sum_{\substack{s \in T \backslash T_{\text{ep}} \\ s \leqslant t}} \sum_{\substack{j \in E_{\text{ep}} \\ j \in E_{(s)}}} x^{\text{ES}}_{j,s} \geqslant 0 \quad \text{for all } t \in T \tag{4.71a}$$

One of the following constraint types has to be added, if $T'_{\text{ep}} \neq T$. Similarly to (4.71a), we can enforce that the first scheduled event in E_{ep} must start at a time $t \in T'_{\text{ep}}$ (4.71b). The same can be done concerning the end of the last scheduled event in E_{ep} (4.71c).

$$\sum_{\substack{s \in T'_{\text{ep}} \\ s \leqslant t}} \sum_{\substack{j \in E_{\text{ep}} \\ j \in E_{(s)}}} |E_{\text{ep}} \cap E_{(s)}| \cdot x^{\text{ES}}_{j,s} - \sum_{\substack{s \in T \backslash T'_{\text{ep}} \\ s \leqslant t}} \sum_{\substack{j \in E_{\text{ep}} \\ j \in E_{(s)}}} x^{\text{ES}}_{j,s} \geqslant 0 \quad \text{for all } t \in T \tag{4.71b}$$

$$\sum_{\substack{s \in T'_{\text{ep}} \\ s \leqslant t}} \sum_{\substack{j \in E_{\text{ep}} \\ j \in E_{(s)}}} |E_{\text{ep}} \cap E_{s}| \cdot x^{\text{ES}}_{j,s+1-d_j} - \sum_{\substack{s \in T \backslash T'_{\text{ep}} \\ s \leqslant t}} \sum_{\substack{j \in E_{\text{ep}} \\ j \in E_{(s)}}} x^{\text{ES}}_{j,s+1-d_j} \geqslant 0 \quad \text{for all } t \in T \tag{4.71c}$$

The case that all scheduled events in E_{ep} must start/end at a time $t \in T'_{\text{ep}}$ is already considered by generating the set J of allowed combinations for event activities.

Timeframe rule: This constraint is already considered by generating the set I of allowed combinations for assignment activities.

Event period zero distance: Let ep be a constraint of this type and let $E_{\text{ep}} \subseteq E$ be the corresponding set of events. The events in E_{ep} have to be adjacent, i. e., their periods have to be pairwise disjoint (4.72) and the duration of their bounding period must be limited by $d_{\max} := \sum_{j \in E_{\text{ep}}} d_j$

(4.73).

$$\sum_{\substack{j\in E_{\text{ep}}\\ j\in E_{(t)}}} \sum_{l\in L_{(j,t)}} x_{j,t,l} \leqslant 1 \qquad \text{for all } t \in T \tag{4.72}$$

$$\text{ES}_j + d_j - \text{ES}_{j'} \leqslant d_{\max} \qquad \text{for all } j, j' \in E_{\text{ep}} \tag{4.73}$$

Event period sequence: Let ep be a constraint of this type and let $E_{\text{ep}} \subseteq E$ be the corresponding set of events and let $(j_1, \cdots, j_{|E_{\text{ep}}|})$ be the events of E_{ep} in the order that is imposed by ep. The starts of their periods are subject to this order.

$$\text{ES}_{j_k} \leqslant \text{ES}_{j_{k'}} + N \cdot (1 - x^X_{j_{k'}}) \qquad \text{for all } k, k' \in [|E_{\text{ep}}|], k < k' \tag{4.74}$$

Scheduling group: Let sg be a constraint of this type and let $E_{\text{sg}} \subseteq E$ be the associated set of events. We further have to consider all intersections of E_{sg} with single event sets that belong to some **Event alternatives** constraint. Let $\mathfrak{E}_{\text{sg}}$ be the system of all these event sets. Similarly, let $\mathcal{E}_{\text{sg}}$ be the system of all event sets, which are intersections of E_{sg} with the union of the event sets of some **Event alternatives** constraint. With $\overline{E}_{\text{sg}} := E_{\text{sg}} \setminus \bigcup_{F \in \mathfrak{E}_{\text{sg}}} F$, we denote the subset of all events in E_{sg} that are not associated with any **Event alternative** constraint. Events of the scheduling groups have to be scheduled mutually together with exceptions concerning event alternatives, formally

- For all $F \in \mathfrak{E}_{\text{sg}} \cup \{\overline{E}_{\text{sg}}\}$ either all or none of the events in F must be scheduled.
- If any event in E_{sg} is scheduled, even every $F \in \mathcal{E}_{\text{sg}} \cup \{\overline{E}_{\text{sg}}\}$ must contain a scheduled event.

which is expressed by (4.75) and (4.76).

$$x^X_j = x^X_{j'} \qquad \text{for all } F \in \mathfrak{E}_{\text{sg}} \cup \{\overline{E}_{\text{sg}}\} \text{ and } j, j' \in F \tag{4.75}$$

$$\sum_{j\in F} |E_{\text{sg}}| \cdot x^X_j \geqslant \sum_{j \in E_{\text{sg}}} x^X_j \qquad \text{for all } F \in \mathcal{E}_{\text{sg}} \cup \{\overline{E}_{\text{sg}}\} \tag{4.76}$$

The second type of constraints modeled by sg concerns assignments. Namely, that assignments must be scheduled if and only if their event is scheduled within an appropriate period. Let $\text{AP}_{\text{sg}} \subseteq A \times 2^T$ be the corresponding set of pairs $(i, \overline{T}_i)$, where $\overline{T}_i$ is the period within the event $j(i)$ of i should start. The desired constraints can be expressed by (4.77) and (4.78) together with

(4.7).

$$y_i^X \geqslant x_{j(i)}^X - \sum_{t \in T_{(i)} \setminus \overline{T}_i} x_{i,t}^{\mathrm{ES}} \quad \text{for all } (i, \overline{T}_i) \in AP_{\mathrm{sg}} \tag{4.77}$$

$$y_i^X \leqslant 1 - \sum_{t \in T_{(i)} \setminus \overline{T}_i} x_{i,t}^{\mathrm{ES}} \quad \text{for all } (i, \overline{T}_i) \in AP_{\mathrm{sg}} \tag{4.78}$$

Qualified resources: This constraint is already considered by generating the set I of allowed combinations for assignment activities.

Resource event capacity: A resource $r \in R$ cannot be assigned to more events than its event capacity c_r^E at any time.

$$\sum_{i \in A_{(t,r)}} \sum_{l \in L_{(i,t,r)}} y_{i,t,l,r} \leqslant c_r^E \quad \text{for all } r \in R, t \in T_{(r)} \tag{4.79}$$

Resource assignment capacity: The allocation of the assignments of a resource $r \in R$ cannot be more than its assignment capacity c_r^A at any time.

$$\sum_{i \in A_{(t,r)}} \sum_{l \in L_{(i,t,r)}} y_{i,t,l,r} \cdot \mathrm{alloc}_i \leqslant c_r^A \quad \text{for all } r \in R, t \in T_{(r)} \tag{4.80}$$

4.12 Remark. The above capacity constraints are simplified versions of the original constraints modeled in [LRS10] since we do not support join opportunity, yet.

Required continuity: Let $A_{\mathrm{rc}} \subseteq A$ be the set of assignments that belong to an constraint of this type. All scheduled assignments in A_{rc} must use the same resource.

$$y_{i,r}^R \leqslant y_{j,r}^R + 2 - y_i^X - y_j^X \quad \text{for all } i, j \in A_{\mathrm{rc}}, r \in R_{(i)} \cap R_{(j)}, \tag{4.81}$$

Discontinuity: Let $A_{\mathrm{dc}}^1 \subseteq A$ and $A_{\mathrm{dc}}^2 \subseteq A$ be the sets of assignments that are given by a constraint of this type. No pair $(i, j) \in A1 \times A2$ may use the same resource.

$$y_{i,r}^R + y_{j,r}^R \leqslant 1 \quad \text{for all } i \in A1_{\mathrm{dc}}, j \in A2_{\mathrm{dc}}, r \in R_{(i)} \cap R_{(j)} \tag{4.82}$$

Resource day: A constraint rd of this type defines limits $\mathrm{maxD}_{\mathrm{rd}}$ on the number of assigned time grid units (actually the corresponding minute values) within working blocks and the length $\mathrm{maxW}_{\mathrm{rd}}$ of working blocks, respectively. The limits may be restricted to certain resources and a certain rule period. Let $R_{\text{rd}} \subseteq R$ and $T_{\text{rd}} \subseteq T$ the sets of resources and time grid points that have to be considered. Maintenance of the limits is expressed by (4.83) and (4.84).

$$\sum_{t \in T_{\text{rd}}} z^{\text{WA}}_{r,j,t} \leqslant \text{maxD}_{\text{rd}} \quad \text{for all } r \in R_{\text{rd}}, j \in [W_r] \tag{4.83}$$

$$\text{WE}_{r,j} - \text{WS}_{r,j} \leqslant \text{maxW}_{\text{rd}} \quad \text{for all } r \in R_{\text{rd}}, j \in [W_r] \tag{4.84}$$

Count duty in periods: Every working rule constraint cd of this type defines a set R_{cd} of resources, the duty of which is measured w.r.t. given locations L_{cd}. Depending on the scheduled location, there may be one or more, say $k_{\text{cd},l}$, time sets $T_{\text{cd},l,k} \subseteq T$, $k = 1, \cdots, k_{\text{cd}}$, $l \in L_{\text{cd}}$, within which the duty has to be measured. Those sets are derived from accordant fix periods or from rolling periods that are suitably represented by cd. Duty can be measured in terms of number of assignments, minutes or resource days, respectively. Concerning the number of assignments, given limits $\min_{\text{cd}}$ and $\max_{\text{cd}}$ are forced by constraints (4.85a). For the minutes case we use (4.85b), where the limits are converted to units of the time grid. The resource days case is handled by (4.85c).

$$\min_{\text{cd}} \leqslant \sum_{l \in L_{\text{cd}}} \sum_{t \in T_{\text{cd},k,l}} \sum_{i \in A} y^{\text{ASLR}}_{i,t,l,r} \leqslant \max_{\text{cd}} \quad \text{for all } r \in R_{\text{cd}}, k \in [k_{\text{cd}}] \tag{4.85a}$$

$$\min_{\text{cd}} \leqslant \sum_{l \in L_{\text{cd}}} \sum_{t \in T_{\text{cd},k,l}} \sum_{i \in A} y_{i,t,l,r} \leqslant \max_{\text{cd}} \quad \text{for all } r \in R_{\text{cd}}, k \in [k_{\text{cd}}] \tag{4.85b}$$

$$\min_{\text{cd}} \leqslant \sum_{l \in L_{\text{cd}}} \sum_{d \in [W_r]} \sum_{i \in A} y^{\text{ADLR}}_{i,d,l,r} \leqslant \max_{\text{cd}} \quad \text{for all } r \in R_{\text{cd}}, k \in [k_{\text{cd}}] \tag{4.85c}$$

Count filter in periods: Let cf be a constraint of this type and R_{cf}, A_{cf} sets of resources and filter assignments that are defined by cf. For all $r \in R_{\text{cf}}$ the times of activity or inactivity are measured w.r.t. the assignments in A_{cf} and rolling periods. Constraint cf further defines if either the total amount of the measured times or the longest consecutive time of activity/inactivity must be within given limits $\min_{\text{cf}}$ and $\max_{\text{cf}}$. From a rule period and rolling periods given by cf, we obtain a system of time sets $M_{\text{cf},1}, \cdots, M_{\text{cf},k_{\text{cf}}}$ that have to be checked. Times must either be measured in terms of minutes (converted to time grid units) or in terms of resource days. Depending on the corresponding time unit requirement of cf, we have $M_k \subseteq T$ or $M_k \subseteq [W_{\text{cf}}]$, respectively. If the total amount is considered, compliance with the given limits is granted by (4.86)(4.87). If consecutive times are desired but only upper bounds are imposed we can use (4.88). But if consecutive times of positive minimum duration are required, we must introduce some more auxiliary variables and use (4.134) of Section 4.5. The variables $\widetilde{y}$ have (for convenience) case dependent assignments as shown in Table 4.1.

Table 4.1. Assignments of $\widetilde{y}$ for MINT constraint **Count filter in periods** depending on the combination of unit and filter mode

parameters		
unit	filter mode	$\tilde{y}$
minute	inclusive	y
resource day	inclusive	y^{ADLR}
minute	exclusive	$1 - y$
resource day	exclusive	$1 - y^{ADLR}$

$$\sum_{t \in M_{\mathrm{cf},k}} \sum_{i \in A_{\mathrm{cf}}} \sum_{l \in L_{(i)}} \tilde{y}_{i,t,l,r} \leqslant \max{}_{\mathrm{cf}} \quad \text{for all } r \in R_{\mathrm{cf}}, k \in [k_{\mathrm{cf}}] \tag{4.86}$$

$$\min{}_{\mathrm{cf}} \leqslant \sum_{t \in M_{\mathrm{cf},k}} (1 - y^{\mathrm{CF3}}_{\mathrm{cf},t} + \sum_{i \in A_{\mathrm{cf}}} \sum_{l \in L_{(i)}} \tilde{y}_{i,t,l,r}) \quad \text{for all } r \in R_{\mathrm{cf}}, k \in [k_{\mathrm{cf}}] \tag{4.87}$$

$$\sum_{i \in A_{\mathrm{cf}}} \sum_{l \in L_{(i)}} \sum_{s=t}^{t+\max_{\mathrm{cf}}} \tilde{y}_{i,t,l,r} \leqslant \max{}_{\mathrm{cf}} \quad \text{for all } r \in R_{\mathrm{cf}}, t \in T \, (t \in W_{\mathrm{cf}}) \tag{4.88}$$

Assignment distance: For the fully supported version of this constraint, see Section 4.5. Our implementation for the given test instances considers only a constraint variation ad that requires for two given sets $A_{\mathrm{ad},1}, A_{\mathrm{ad},2} \subseteq A$ of assignments that for every pair of assignments $(i,j) \in A_{\mathrm{ad},1} \times A_{\mathrm{ad},2}$ the resource day of j is at least $\min_{\mathrm{ad}} - 1$ days later than the resource day of i.

$$\mathrm{AD}_j \geqslant \mathrm{AD}_i + \min{}_{\mathrm{ad}} - 1 - W_{\mathrm{ad}}(2 - y_i^X - y_j^X) \quad \text{for all } i \in A_{\mathrm{ad},1}, j \in A_{\mathrm{ad},2}.$$

Resource day sequence: Let rds be a constraint of this type and let A_{rds} be the corresponding set of assignments. If ranges $\{\delta^{\min}_{\text{rds},k}, \delta^{\max}_{\text{rds},k}\}$, $k \in [|A_{\text{rds}}|]$, are given, the gaps between subsequent active resource days for assignments in A_{rds} must comply with these ranges (4.89)(4.90)

$$\text{SD}_{k+1} \geqslant \text{SD}_k + (1 - y^{\text{SD}}_{W_{\text{sd}}+1,k+1}) \cdot \delta^{\min}_{\text{rds},k} \quad \text{for all } k < |A_{\text{rds}}| \tag{4.89}$$

$$\text{SD}_{k+1} \leqslant \text{SD}_k + \delta^{\max}_{\text{rds},k} \quad \text{for all } k < |A_{\text{rds}}| \tag{4.90}$$

On the other hand, if no δ values are specified, assignments must consecutively use the allowed weekdays. Let T_{rds} the set of all time grid points of the planning horizon at which an allowed weekday begins. Then, for every $t \in T_{\text{rds}}$ which lies between the start of the first scheduled assignment of A_{rds} and the end of the last scheduled assignment in A_{rds}, there must be an active assignment for r_{rds} in $\{t, \cdots, t + n^{\text{day}} - 1\}$:

$$y^{\text{RDS3}}_{\text{rds},t} - \sum_{i \in A_{\text{rds}}} \sum_{l \in L_{(i)}} \sum_{s=t}^{t+n^{\text{day}}-1} y_{i,l,r_{\text{rds}},s} \leqslant 0 \tag{4.91}$$

Resource day assignments: All scheduled assignments A_{rda} of a constraint rda of this type must be scheduled at the same resource day of the same resource.

$$y^{\text{AD}}_{i,d} \leqslant y^{\text{AD}}_{j,d} + 2 - y^X_i - y^X_j \quad \text{for all } i, j \in A_{\text{rda}}, d \in W_{\text{ad}} \tag{4.92}$$

4.3 EA Framework

The EA framework for the MINT SP was developed and implemented by Torben Rabe and is the topic of his Diploma Thesis [Rab12]. The essence of Rabes work is the development of an encoding function (schedule generation scheme, SGS) that considers precedence relations as well as all other restrictions that are required within the couple of test cases provided by the MINT company. His encoding function can be understood as a sophisticated extension of the SGS for the RCPSP (see Example 4.1 of subsection 4.1.2) described in [Har02]. Similar to the RCPSP, the MINT SP imposes precedence relations (given by **Event period sequence** rules) that provide the opportunity to update a set of eligible events in a sequential scheduling procedure. Precalculations that take **Event period zero distance** and **Assignment distance** constraints into account allow for a reduction of the search space by treating sets $\mathcal{E} \subseteq E$ of events at once. Considering such event sets as top level objects, Rabes SGS seeks for the earliest feasible starting times. This principle is similar to that of the heuristic for the TSP with time windows problem in Section 2.4. For the TSPTW problem, this task could still be managed by a single case differentiating procedure. There, the procedure is called within the scheduling steps in Algorithm 2, line 22 of the exact method and item 7 of the heuristic, respectively. However, for the MINT SP we have to consider a more complex system of constraints in order to find the first feasible points in time. Rabe treats the constraints hierarchically by the definition of corresponding functions. The hierarchy is given by the object types of the scheduling problem. Top level is the precalculated event sets, followed by single events, assignments for events and, finally, qualified resources for assignments. Let us write $f_{c,i}^{\text{type}}$ to denote a function that returns the first feasible starting time for an object type w.r.t. some applicable instant constraint i of a certain MINT constraint type c. Clearly, in addition to some point in time the functions require the instance information that concerns the considered constraints (see subsection 4.1.1) as input parameters. Overriding the associated method of an abstract JAVA class, a function that belongs to a certain level in the hierarchy can be defined by reverting to conjunctions of others. Herein, conjunction usually means to determine the maximum value of either all applicable constraint types for the object under consideration or necessary constraints that concern sub-level objects of the considered object. Algorithm 6 sketches an implementation of this principle. The resource level requires record keeping of the feasible time slots of the resources.

Arguments of Rabes schedule generation scheme are random key genotypes the EA operates on. Random keys are random numbers in $[0, 1]$ and are often used to represent the order of the objects in an actual solution of a hard constrained problem (see, e. g., [YG10], Subsections 7.3.3.3 and 7.4.3.1). One can simply use general real-coded EAs (see subsection 1.5.4) in order to deal with random key genotypes. Random keys may also represent operational order preferences of a planning heuristic that is used within the encoding function of a hybrid EA. The latter possibility was chosen for the MINT SP, where the genotypes represent the order in which the SGS tries candidate objects that are about to be scheduled.

Algorithm 6: Recursive definition of first feasible function

```
input  : $t \in T$, instance and partial planning information
output : first feasible $s \in T$
1 let ot be the (sub-level) object type that must be considered;
2 let $C$ be the set of MINT constraint types that must be considered;
3 for each $c \in C$ let $I_c$ be the set of constraints specified by instance;
4 while $s \neq t$ do
5     $s = t$;
6     for $c \in C$ do
7         for $i \in I_c$ do $t = f^{\text{ot}}_{c,i}(t)$;
8 return $s$;
```

4.4 Experiments

Based on the new model [LRS10] our corporation partner MINT generated 14 test instances for which the actual planning algorithm TMS faces difficulties. Table 4.2 gives an overview about the properties of these instances. Table 4.3 presents algorithmic results. The ILP experiments were performed on an Intel Atom processor with 1.6 GHz clock rate and 512 MB RAM while the EA experiments ran on an Intel Pentium Dual Core machine with 2.1 GHz clock rate and 4 GB RAM (here, times are given per SGS). For the EA, a population size of 200 individuals was chosen. We ran 10 EA trials for every instance. The detection of a feasible solution was chosen as stopping criterion. We always state the average number of EA generations. The check mark ✓ means that the algorithm terminated with a solution. For the ILP, ✓ means that the feasibility of a given solution could be verified.

We see, that the EA is able to solve most of these rather small test cases very fast and within one generation. The genetic operators take effect only for three instances, the size and complexity of which requires a corresponding longer running time being proportional to the product of the population size, the required number of generations and the required time per SGS.

Imposing a time limit of 10 minutes, only four instances could be solved by the ILP (using CPLEX) proving both the feasibility and optimality of the corresponding EA solutions. For four another instances the ILP was able to

Table 4.2. Test instances for the MINT SP. Each instance is given along with the number of events, assignments and resources ($|E|$,$|A|$ and $|R|$), respectively.

Instance Name	Parameters		
	$\lvert E\rvert$	$\lvert A\rvert$	$\lvert R\rvert$
Continuities	57	198	12
EighteenClassesFiveSlots	144	216	21
Flexible	360	360	4
FlexibleForceSkipDay	5	6	2
GreedyDayDeadEnd	2	4	3
GreedyStartDeadEnd	2	4	3
Range1	4	4	4
Range2	4	4	1
RequirementMultiDay	9	27	3
RequirementOneDay	3	9	3
Sequential	360	360	4
SixClassesFiveSlots	48	72	7
ThreeClassesTwoSlots	12	18	4
Weekday	360	360	4

Table 4.3. Performance results concerning the MINT SP test instances.

Instance Name	ILP		Heuristic		
		Sec.		Sec./SGS	∅Gen.
Continuities	-	-	✓	0.150	9
EighteenClassesFiveSlots	-	-	✓	0.061	78.3
Flexible	-	-	✓	0.150	1
FlexibleForceSkipDay	✓	>600.0	✓	0.010	1
GreedyDayDeadEnd	✓	0.5	✓	0.015	1
GreedyStartDeadEnd	✓	2.0	✓	0.017	1
Range1	✓	415.1	✓	0.020	1
Range2	✓	>600.0	✓	0.024	1
RequirementMultiDay	✓	>600.0	✓	0.050	1
RequirementOneDay	✓	42.2	✓	0.016	1
Sequential	-	-	✓	1.600	1
SixClassesFiveSlots	-	-	✓	0.024	16.9
ThreeClassesTwoSlots	✓	>600.0	✓	0.020	1
Weekday	-	-	✓	1.800	1

verify the feasibility of EA solutions after being provided with them. Since the current ILP implementation does no preprocessing in order to reduce the number of decision variables (see subsection 4.2.1), it already requires too much memory for the six larger test instances. Such preprocessing is also supposed to improve the performance of the ILP. However, concerning large real-world problems, only partial ILP formulations are intended to be used as integral parts of LP-relaxation based heuristics.

4.5 Further ILP Constraints

4.5.1 Further Auxiliary Variables

In order to handle mobile resources we introduce the variables

- $\lambda_{r,t,l} \in \{0,1\}$, $r \in R, t \in T_{(r)}, l \in L_{(t,r)}$
  $\lambda_{r,t,l}$ is 1, if and only if resource r is assigned at time t and location l.
- $\overline{\lambda}_{r,t,l,l'} \in \{0,1\}$, $r \in R, t \in T_{(r)}, l \in L_{(t,r)}, l' \in L_{(s_t,r)}$
  $\overline{\lambda}_{r,t,l,l'}$ is 1, if and only if resource r has location l at time t and location l' at time s_t being is the earliest time in $T_{(r)}$ that is greater than t.

Let cf be a **Count filter in periods** constraint and R_{cf} and A_{cf} be the corresponding sets of resources and assignments, respectively. If cf concerns consecutive times of minimum length $\min_{\text{cf}} > 1$, we need the following indicators.

- $z^{\text{cons}}_{\text{cf},r,t} \in \{0,1\}$, $r \in R_{\text{cf}}$, $t \in T$
  $z^{\text{cons}}_{\text{cf},r,t}$ is 1, if and only if r is active (inactive) w.r.t. assignments in A_{cf} for every $s \in [t, t + \min_{\text{cf}} - 1]$
- $z^{\text{consD}}_{\text{cf},r,d} \in \{0,1\}$, $r \in R_{\text{cf}}$, $d \in [W_r]$
  $z^{\text{consD}}_{\text{cf},r,d}$ is 1, if and only if r is active (inactive) w.r.t. assignments in A_{cf} at every resource day $d' \in [d, d + \min_{\text{cf}} - 1]$

Let ad be an **Assignment distance** rule and let $A_{\text{ad},1}, A_{\text{ad},2} \subseteq A$ be the corresponding sets of assignments. We need to introduce a couple of additional auxiliary variables in order to model ad

- $\text{LS}_{\text{ad},k} \in T$, $k \in \{1,2\}$
  $\text{LS}_{\text{ad},k}$ is the earliest start time of all assignments in $A_{\text{ad},k}$.

- $y^{\mathrm{LS}}_{\mathrm{ad},k,t} \in \{0,1\}$, $k \in \{1,2\}$, $t \in T$
  $y^{\mathrm{LS}}_{\mathrm{ad},k,t}$ is 1, if and only if $\mathrm{LS}_{\mathrm{ad},k}$ is t.
- $\mathrm{LE}_{\mathrm{ad},k} \in T$, $k \in \{1,2\}$
  $\mathrm{LE}_{\mathrm{ad},k}$ is the latest finish time of all assignments in $A_{\mathrm{ad},k}$.
- $y^{\mathrm{LE}}_{\mathrm{ad},k,t} \in \{0,1\}$, $k \in \{1,2\}$, $t \in T$
  $y^{\mathrm{LE}}_{\mathrm{ad},k,t}$ is 1, if and only if $\mathrm{LE}_{\mathrm{ad},k}$ is t.
- $\mathrm{LSD}_{\mathrm{ad},k} \in W_{\mathrm{ad}}$, $k \in \{1,2\}$
  $\mathrm{LSD}_{\mathrm{ad},k}$ is the resource day of the earliest assignment in $A_{\mathrm{ad},k}$.
- $y^{\mathrm{LSD}}_{\mathrm{ad},k,d} \in \{0,1\}$, $k \in \{1,2\}$, $d \in [W_{\mathrm{ad}}]$
  $y^{\mathrm{LSD}}_{\mathrm{ad},k,d}$ is 1, if and only if $\mathrm{LSD}_{\mathrm{ad},k}$ is d.
- $\mathrm{LED}_{\mathrm{ad},k} \in W_{\mathrm{ad}}$, $k \in \{1,2\}$
  $\mathrm{LED}_{\mathrm{ad},k}$ is the resource day of the latest assignment in $A_{\mathrm{ad},k}$.
- $y^{\mathrm{LED}}_{\mathrm{ad},k,d} \in \{0,1\}$, $k \in \{1,2\}$, $d \in [W_{\mathrm{ad}}]$
  $y^{\mathrm{LED}}_{\mathrm{ad},k,d}$ is 1, if and only if $\mathrm{LED}_{\mathrm{ad},k}$ is d.
- $\mathrm{GS}_{\mathrm{ad}} \in T$
  $\mathrm{GS}_{\mathrm{ad}}$ is the beginning of the gap between both assignment lists, $A_{\mathrm{ad},1}$ and $A_{\mathrm{ad},2}$.
- $y^{\mathrm{GS}}_{\mathrm{ad},t} \in \{0,1\}$, $t \in T$
  $y^{\mathrm{GS}}_{\mathrm{ad},t}$ is 1, if and only if $\mathrm{GS}_{\mathrm{ad}}$ is t.
- $\mathrm{GE}_{\mathrm{ad}} \in T$
  $\mathrm{GE}_{\mathrm{ad}}$ is the end of the gap between both assignment lists, $A_{\mathrm{ad},1}$ and $A_{\mathrm{ad},2}$.
- $y^{\mathrm{GE}}_{\mathrm{ad},t} \in \{0,1\}$, $t \in T$
  $y^{\mathrm{GE}}_{\mathrm{ad},t}$ is 1, if and only if $\mathrm{GE}_{\mathrm{ad}}$ is t.
- $\mathrm{GSD}_{\mathrm{ad}} \in [W_{\mathrm{ad}}]$
  $\mathrm{GSD}_{\mathrm{ad}}$ is the first resource day in the gap between both assignment lists, $A_{\mathrm{ad},1}$ and $A_{\mathrm{ad},2}$.
- $y^{\mathrm{GSD}}_{\mathrm{ad},d} \in \{0,1\}$, $d \in [W_{\mathrm{ad}}]$
  $y^{\mathrm{GSD}}_{\mathrm{ad},d}$ is 1, if and only if $\mathrm{GSD}_{\mathrm{ad}}$ is d.
- $\mathrm{GED}_{\mathrm{ad}} \in [W_{\mathrm{ad}}]$
  $\mathrm{GED}_{\mathrm{ad}}$ is the last resource day in the gap between both assignment lists,

$A_{\text{ad},1}$ and $A_{\text{ad},2}$.

- $y^{\text{GED}}_{\text{ad},d} \in \{0,1\}$, $d \in [W_{\text{ad}}]$
  $y^{\text{GED}}_{\text{ad},d}$ is 1, if and only if GED_{ad} is d.

If the constraint ad requests a minimal gap, we further need variables that indicate if a time $t \in T$ within the gap should be disallowed for every assignment that starts directly after the gap. Defining plus and minus by

$$\text{plus} = \begin{cases} \text{min}_{\text{ad}}, & \text{if unit is minutes} \\ n^{\text{day}} \text{min}_{\text{ad}} + \frac{n^{\text{turn}}}{2}, & \text{if unit is resource days} \end{cases},$$

$$\text{minus} = \begin{cases} 0, & \text{if unit is minutes} \\ \frac{n^{\text{turn}}}{2}, & \text{if unit is resource days} \end{cases},$$

those indicators are given by

- $y^{U}_{\text{ad},t} \in \{0,1\}$, $t \in T$
  $y^{U}_{\text{ad},t}$ is 1, if and only if $t \in U = [\text{GS}_{\text{ad}} + \text{plus}, \text{GE}_{\text{ad}} - \text{minus}]$

4.5.2 Further Relationships between Variables

Every resource has exactly one location at one point in time (4.93). A resource has location l at time t if it is used by an active assignment at time t and location l (4.94).

$$\sum_{l \in L_{(t,r)}} \lambda_{r,t,l} = 1 \quad \text{for all } r \in R, t \in T_{(r)} \tag{4.93}$$

$$\lambda_{r,t,l} \geqslant \frac{1}{c^E_r} \cdot \sum_{i \in A_{(t,l,r)}} y_{i,t,l,r} \quad \text{for all } r \in R, t \in T_{(r)}, l \in L_{(t,r)} \tag{4.94}$$

The meaning of variables $\overline{\lambda}_{r,t,l,l'}$ is expressed as follows

$$2 \cdot \overline{\lambda}_{r,t,l,l'} \leqslant \lambda_{r,t,l} + \lambda_{r,s_t,l'} \quad \text{for all } r \in R, t \in T_{(r)}, l \in L_{(t,r)}, l' \in L_{(s_t,r)} \tag{4.95}$$

$$\sum_{l' \in L_{(s_t,r)}} \overline{\lambda}_{r,t,l,l'} = \lambda_{r,t,l} \quad \text{for all } r \in R, t \in T_{(r)}, l \in L_{(t,r)} \tag{4.96}$$

Whenever a resource changes its location, we want it to be immediately active upon arrive. This convention will be used for the formulation of MINT constraints **Allow non-local resources** and **Travel time**.

$$\sum_{\substack{l,l' \in L_{(r)} \\ l \neq l'}} \lambda_{r,t,l,l'} \leqslant z_{r,s_t} \quad \text{for all } r \in R, t \in T_{(r)}. \tag{4.97}$$

The auxiliary variables that concern assignment lists (given by **Assignment distance** rules ad) are derived from the corresponding variables for single assignments. A unique start time of the bounding period of an assignment list exists, if and only if it contains a scheduled assignment (4.98a)(4.99a). The relationship between start time and the corresponding indicators is given by (4.100a). A time that isn't start time of any assignment in a list can't be start time of its bounding period (4.101a) and the start time of the bounding period must be less or equal to the start time of every assignment in the list (4.102a). Similar conditions must hold for the finish time of a bounding period (4.103a)-(4.107a), the resource day at which a bounding period starts (4.98b)-(4.102b) and the resource day at which a bounding period ends (4.103b)-(4.107b).

$$\sum_{t \in T} y^{\text{LS}}_{\text{ad},k,t} \leqslant \sum_{i \in A_{\text{ad},k}} y^X_i \quad \text{for all } k \in \{1,2\} \tag{4.98a}$$

$$\sum_{d \in [W_{\text{ad}}]} y^{\text{LSD}}_{\text{ad},k,d} \leqslant \sum_{i \in A_{\text{ad},k}} y^X_i \quad \text{for all } k \in \{1,2\} \tag{4.98b}$$

$$\sum_{t \in T} y^{\text{LS}}_{\text{ad},k,t} \geqslant y^X_i \quad \text{for all } k \in \{1,2\}, i \in A_{\text{ad},k} \tag{4.99a}$$

$$\sum_{d \in [W_{\text{ad}}]} y^{\text{LSD}}_{\text{ad},k,d} \geqslant y^X_i \quad \text{for all } k \in \{1,2\}, i \in A_{\text{ad},k} \tag{4.99b}$$

$$\text{LS}_{\text{ad},k} = \sum_{t \in T} t \cdot y^{\text{LS}}_{\text{ad},k,t} \quad \text{for all } k \in \{1,2\} \tag{4.100a}$$

$$\text{LSD}_{\text{ad},k} = \sum_{d \in [W_{\text{ad}}]} d \cdot y^{\text{LS}}_{\text{ad},k,d} \quad \text{for all } k \in \{1,2\} \tag{4.100b}$$

$$y^{\mathrm{LS}}_{\mathrm{ad},k,t} \leqslant y^{\mathrm{AS}}_{i,t} \quad \text{for all } k \in \{\}, i \in A_{\mathrm{ad},k}, t \in T_{(i)} \tag{4.101a}$$
$$y^{\mathrm{LSD}}_{\mathrm{ad},k,d} \leqslant y^{\mathrm{AD}}_{i,d} \quad \text{for all } k \in \{\}, i \in A_{\mathrm{ad},k}, d \in [W_{\mathrm{ad}}] \tag{4.101b}$$

$$\mathrm{LS}_{\mathrm{ad},k} \leqslant \mathrm{AS}_i \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k} \tag{4.102a}$$
$$\mathrm{LSD}_{\mathrm{ad},k} \leqslant \mathrm{AD}_i \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k} \tag{4.102b}$$

$$\sum_{t\in T} y^{\mathrm{LE}}_{\mathrm{ad},k,t} \leqslant \sum_{i\in A_{\mathrm{ad},k}} y^X_i \quad \text{for all } k \in \{1,2\} \tag{4.103a}$$
$$\sum_{d\in[W_{\mathrm{ad}}]} y^{\mathrm{LED}}_{\mathrm{ad},k,d} \leqslant \sum_{i\in A_{\mathrm{ad},k}} y^X_i \quad \text{for all } k \in \{1,2\} \tag{4.103b}$$

$$\sum_{t\in T} y^{\mathrm{LE}}_{\mathrm{ad},k,t} \geqslant y^X_i \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k} \tag{4.104a}$$
$$\sum_{d\in[W_{\mathrm{ad}}]} y^{\mathrm{LED}}_{\mathrm{ad},k,d} \geqslant y^X_i \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k} \tag{4.104b}$$

$$\mathrm{LE}_{\mathrm{ad},k} = \sum_{t\in T} t \cdot y^{\mathrm{LE}}_{\mathrm{ad},k,t} \quad \text{for all } k \in \{1,2\} \tag{4.105a}$$
$$\mathrm{LED}_{\mathrm{ad},k} = \sum_{d\in[W_{\mathrm{ad}}]} d \cdot y^{\mathrm{LS}}_{\mathrm{ad},k,d} \quad \text{for all } k \in \{1,2\} \tag{4.105b}$$

$$y^{\mathrm{LE}}_{\mathrm{ad},k,t} \leqslant y^{\mathrm{AS}}_{i,t+1-\overline{d}_i} \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k}, t \in T_{(i)} \tag{4.106a}$$
$$y^{\mathrm{LED}}_{\mathrm{ad},k,d} \leqslant y^{\mathrm{AD}}_{i,d} \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k}, d \in [W_{\mathrm{ad}}] \tag{4.106b}$$

$$\mathrm{LE}_{\mathrm{ad},k} \geqslant \mathrm{AS}_i + \overline{d}_i - 1 \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k} \tag{4.107a}$$
$$\mathrm{LED}_{\mathrm{ad},k} \leqslant \mathrm{AD}_i \quad \text{for all } k \in \{1,2\}, i \in A_{\mathrm{ad},k} \tag{4.107b}$$

Note, that all time variables of an assignment list are 0, if the list does not contain any scheduled assignment.

The gap has unique bounds, if and only if both assignment lists contain a scheduled assignment, which is the case, if they both have a unique start time (4.108a)-(4.111b)

$$\sum_{t \in T} y^{\text{GS}}_{\text{ad},t} \geqslant \sum_{t \in T} (y^{\text{LS}}_{\text{ad},1,t} + y^{\text{LS}}_{\text{ad},2,t}) - 1 \tag{4.108a}$$

$$\sum_{d \in [W_{\text{ad}}]} y^{\text{GSD}}_{\text{ad},d} \geqslant \sum_{d \in [W_{\text{ad}}]} (y^{\text{LSD}}_{\text{ad},1,d} + y^{\text{LSD}}_{\text{ad},2,d}) - 1 \tag{4.108b}$$

$$\sum_{t \in T} y^{\text{GS}}_{\text{ad},t} \leqslant \sum_{t \in T} y^{\text{LS}}_{\text{ad},k,t} \quad \text{for all } k \in \{1,2\} \tag{4.109a}$$

$$\sum_{d \in [W_{\text{ad}}]} y^{\text{GSD}}_{\text{ad},d} \leqslant \sum_{d \in [W_{\text{ad}}]} y^{\text{LSD}}_{\text{ad},k,d} \quad \text{for all } k \in \{1,2\} \tag{4.109b}$$

$$\sum_{t \in T} y^{\text{GE}}_{\text{ad},t} \geqslant \sum_{t \in T} (y^{\text{LS}}_{\text{ad},1,t} + y^{\text{LS}}_{\text{ad},2,t}) - 1 \tag{4.110a}$$

$$\sum_{d \in [W_{\text{ad}}]} y^{\text{GED}}_{\text{ad},d} \geqslant \sum_{d \in [W_{\text{ad}}]} (y^{\text{LSD}}_{\text{ad},1,d} + y^{\text{LSD}}_{\text{ad},2,d}) - 1 \tag{4.110b}$$

$$\sum_{t \in T} y^{\text{GE}}_{\text{ad},t} \leqslant \sum_{t \in T} y^{\text{LS}}_{\text{ad},k,t} \quad \text{for all } k \in \{1,2\} \tag{4.111a}$$

$$\sum_{d \in [W_{\text{ad}}]} y^{\text{GED}}_{\text{ad},d} \leqslant \sum_{d \in [W_{\text{ad}}]} y^{\text{LSD}}_{\text{ad},k,d} \quad \text{for all } k \in \{1,2\} \tag{4.111b}$$

The relationships between the gap bounds and the corresponding indicators are as usual (4.112a)-(4.113b).

$$\text{GS}_{\text{ad}} = \sum_{t \in T} t \cdot y^{\text{GS}}_{\text{ad},t} \tag{4.112a}$$

$$\text{GSD}_{\text{ad}} = \sum_{d \in [W_{\text{ad}}]} d \cdot y^{\text{GSD}}_{\text{ad},d} \tag{4.112b}$$

$$\text{GE}_{\text{ad}} = \sum_{t \in T} t \cdot y^{\text{GE}}_{\text{ad},t} \tag{4.113a}$$

$$\mathrm{GED}_{\mathrm{ad}} = \sum_{d \in [W_{\mathrm{ad}}]} d \cdot y^{\mathrm{GED}}_{\mathrm{ad},d} \tag{4.113b}$$

The gap start time is $\min(\mathrm{LE}_{\mathrm{ad},1}, \mathrm{LE}_{\mathrm{ad},2}) + 1$ (4.114a)(4.115a) and the first resource day of the gap is $\min(\mathrm{LED}_{\mathrm{ad},1}, \mathrm{LED}_{\mathrm{ad},2}) + 1$ (4.114b)(4.115b).

$$\mathrm{GS}_{\mathrm{ad}} \leqslant \mathrm{LE}_{\mathrm{ad},k} + 1 \quad \text{for all } k \in \{1,2\} \tag{4.114a}$$

$$\mathrm{GSD}_{\mathrm{ad}} \leqslant \mathrm{LED}_{\mathrm{ad},k} + 1 \quad \text{for all } k \in \{1,2\} \tag{4.114b}$$

$$\sum_{s=1}^{t+1} y^{\mathrm{GS}}_{\mathrm{ad},s} \leqslant \sum_{s=1}^{t} (y^{\mathrm{LE}}_{\mathrm{ad},1,s} + y^{\mathrm{LE}}_{\mathrm{ad},2,s}) \quad \text{for all } t \in T \tag{4.115a}$$

$$\sum_{s=1}^{d+1} y^{\mathrm{GSD}}_{\mathrm{ad},s} \leqslant \sum_{s=1}^{d} (y^{\mathrm{LED}}_{\mathrm{ad},1,s} + y^{\mathrm{LED}}_{\mathrm{ad},2,s}) \quad \text{for all } d \in [W_{\mathrm{ad}}] \tag{4.115b}$$

The gap end time is $\max(\mathrm{LS}_{\mathrm{ad},1}, \mathrm{LS}_{\mathrm{ad},2}) - 1$ (4.116a)(4.117a) and the last day of the gap is $\max(\mathrm{LSD}_{\mathrm{ad},1}, \mathrm{LSD}_{\mathrm{ad},2}) - 1$ (4.116b)(4.117b).

$$\mathrm{GE}_{\mathrm{ad}} \geqslant \mathrm{LS}_{\mathrm{ad},k} - 1 \quad \text{for all } k \in \{1,2\} \tag{4.116a}$$

$$\mathrm{GED}_{\mathrm{ad}} \geqslant \mathrm{LSD}_{\mathrm{ad},k} - 1 \quad \text{for all } k \in \{1,2\} \tag{4.116b}$$

$$\sum_{s=t-1}^{N} y^{\mathrm{GE}}_{\mathrm{ad},s} \leqslant \sum_{s=t}^{N} (y^{\mathrm{LS}}_{\mathrm{ad},1,s} + y^{\mathrm{LS}}_{\mathrm{ad},2,s}) \quad \text{for all } t \in T \tag{4.117a}$$

$$\sum_{s=d-1}^{W_{\mathrm{ad}}} y^{\mathrm{GED}}_{\mathrm{ad},s} \leqslant \sum_{s=d}^{W_{\mathrm{ad}}} (y^{\mathrm{LSD}}_{\mathrm{ad},1,s} + y^{\mathrm{LSD}}_{\mathrm{ad},2,s}) \quad \text{for all } d \in [W_{\mathrm{ad}}] \tag{4.117b}$$

The variables that indicate if times $t \in T$ are in the interval $U = [\mathrm{GS} + \mathrm{plus}, \mathrm{GE} - \mathrm{minus}]$ of times that should be unallowed by timeframe rules for each assignment that starts directly after the end of the gap (minimalgap case) are given by (4.118)(4.119)(4.120).

$$y^{U}_{\mathrm{ad},t} \leqslant \sum_{s=t}^{N-\mathrm{minus}} y^{\mathrm{GE}}_{\mathrm{ad},s+\mathrm{minus}} \quad \text{for all } t \in T \tag{4.118}$$

$$y^U_{\mathrm{ad},t} \leqslant \sum_{s=1+\mathrm{plus}}^{t} y^{\mathrm{GS}}_{\mathrm{ad},s-\mathrm{plus}} \quad \text{for all } t \in T \tag{4.119}$$

$$1 + y^U_{\mathrm{ad},t} \geqslant \sum_{s=1+\mathrm{plus}}^{t} y^{\mathrm{GS}}_{\mathrm{ad},s-\mathrm{plus}} + \sum_{s=t}^{N-\mathrm{minus}} y^{\mathrm{GE}}_{\mathrm{ad},s+\mathrm{minus}} \quad \text{for all } t \in T \tag{4.120}$$

We derive the additional indicators that are required by working rules cf of the **Count filter in periods** type for the case that the length of consecutive activities must at least be $\min_{\mathrm{cf}} > 1$. The case that counting units are minutes (converted to time grid units) is expressed by (a) and (b) while (c) and (d) deal with resource days. Times of activity are considered by (a) and (c), times of inactivity by (b) and (d).

$$z^{\mathrm{cons}}_{\mathrm{w},r,t} \geqslant 1 - \frac{1}{\min_{\mathrm{cf}}} \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{s=t}^{t+\min_{\mathrm{cf}}-1} y_{i,s,l,r} \quad \text{for all } r \in R_{\mathrm{w}}, t \in T \tag{4.121a}$$

$$z^{\mathrm{cons}}_{\mathrm{w},r,t} \geqslant 1 - \frac{1}{\min_{\mathrm{cf}}} \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{s=t}^{t+\min_{\mathrm{cf}}-1} (1 - y_{i,s,l,r}) \quad \text{for all } r \in R_{\mathrm{w}}, t \in T \tag{4.121b}$$

$$z^{\mathrm{consD}}_{\mathrm{w},r,d} \geqslant 1 - \frac{1}{\min_{\mathrm{cf}}} \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{d'=d}^{d+\min_{\mathrm{cf}}-1} y^{ADLR}_{i,d',l,r} \quad \text{for all } r \in R_{\mathrm{w}}, d \in [W_r] \tag{4.121c}$$

$$z^{\mathrm{consD}}_{\mathrm{w},r,d} \geqslant 1 - \frac{1}{\min_{\mathrm{cf}}} \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{d'=d}^{d+\min_{\mathrm{cf}}-1} (1 - y^{ADLR}_{i,d',l,r}) \quad \text{for all } r \in R_{\mathrm{w}}, d \in [W_r] \tag{4.121d}$$

$$\min_{\mathrm{cf}} \cdot z^{\mathrm{cons}}_{\mathrm{w},r,t} \leqslant \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{s=t}^{t+\min_{\mathrm{cf}}-1} y_{i,s,l,r} \quad \text{for all } r \in R_{\mathrm{w}}, t \in T \tag{4.122a}$$

$$\min_{\mathrm{cf}} \cdot z^{\mathrm{cons}}_{\mathrm{w},r,t} \leqslant \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{s=t}^{t+\min_{\mathrm{cf}}-1} (1 - y_{i,s,l,r}) \quad \text{for all } r \in R_{\mathrm{w}}, t \in T \tag{4.122b}$$

$$\min_{\mathrm{cf}} \cdot z^{\mathrm{consD}}_{\mathrm{w},r,d} \leqslant \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{d'=d}^{d+\min_{\mathrm{cf}}-1} y^{ADLR}_{i,d',l,r} \quad \text{for all } r \in R_{\mathrm{w}}, d \in [W_r] \tag{4.122c}$$

$$\min_{\mathrm{cf}} \cdot z^{\mathrm{consD}}_{\mathrm{w},r,d} \leqslant \sum_{i \in A_{\mathrm{w}}} \sum_{l \in L_{(i)}} \sum_{d'=d}^{d+\min_{\mathrm{cf}}-1} (1 - y^{ADLR}_{i,d',l,r}) \quad \text{for all } r \in R_{\mathrm{w}}, d \in [W_r] \tag{4.122d}$$

4.5.3 Further Constraints of the MINT SP

Event period minimum distance: Let ep be a constraint of this type and let $E_{\mathrm{ep}} \subseteq E$ be the corresponding set of events. The events in E_{ep} must have disjoint periods (4.123). Moreover, all but the earliest scheduled event in E_{ep} must use the earliest allowed start time with respect to their order. This may be expressed as follows. If some event $j \in E_{\mathrm{ep}}$ takes place at some location l and starts at some time t and other events in E_{ep} start before t, we can calculate the latest allowed start time $s(j,t,l)$ of event j at location l that is before t and must ensure that another event $j' \in E_{\mathrm{ep}}$ is active at time $s(j,t,l)$ (4.124).

$$\sum_{\substack{j \in E_{\mathrm{ep}} \\ j \in E_{(t)}}} \sum_{l \in L_{(j,t)}} x_{j,t,l} \leqslant 1 \quad \text{for all } t \in T \tag{4.123}$$

$$\sum_{j' \in E_{\mathrm{ep}}} x_{j,s(j,t,l),l} \geqslant x^{\mathrm{ES}}_{j,t} + x^{L}_{l} + \frac{1}{|E_{\mathrm{ep}}|} \sum_{t'<t} \sum_{j' \in E_{\mathrm{ep}}} x^{\mathrm{ES}}_{j',t'} - 2$$
$$\text{for all } j \in E_{\mathrm{ep}}, l \in L_{(j)}, t \in T_{(j,l)} \tag{4.124}$$

Assignments running time: Let ar be a constraint of this type and A_{ar} be the accordant set of assignments. By ar, limits $\min_{\mathrm{ar}}$ and $\max_{\mathrm{ar}}$ on the duration of the bounding period w.r.t A_{ar} are either given in terms of time grid units (converted from minute values) or in terms of resource days. Constraints (4.125a) and (4.126a) enforce the required limits in the time grid unit case. For given $i \in A_{\text{ar}}$ and $t \in T_{(i)}$ we use the set $T_{\text{ar}}^{i,t} := T_{(i)} \cap (\{1, \cdots, t-1\} \cup \{t + \min_{\text{ar}}, \cdots, N\})$ in (4.125a). Constraints (4.125b) and (4.126b) do the same in the resource days case.

$$\sum_{s \in T_{\text{ar}}^{i,t}} y_{i,s}^{A} \geqslant 1 \quad \text{for all } i \in A_{\text{ar}}, t \in T_{(i)} \text{ with } T_{\text{ar}}^{i,t} \neq \varnothing \tag{4.125a}$$

$$\sum_{s=1}^{t-1} y_{i,s}^{\text{AD}} + \sum_{s=t+\min_{\text{ar}}}^{W} y_{i,s}^{\text{AD}} \geqslant 1 \quad \text{for all } i \in A_{\text{ar}}, t \in [W + 1 - \min_{\text{ar}}] \tag{4.125b}$$

$$\text{AS}_i + \overline{d}_i + 1 - \text{AS}_j \leqslant \max_{\text{ar}} \quad \text{for all } i, j \in A_{\text{ar}} \tag{4.126a}$$

$$\text{AD}_i + 1 - \text{AD}_j \leqslant \max_{\text{ar}} \quad \text{for all } i, j \in A_{\text{ar}} \tag{4.126b}$$

Event alternatives: Let ea be a constraint of this type. Let $\mathfrak{E}_{\text{ea}}$ be the corresponding system of subsets of E that represents the event alternatives and let $E_{\text{ea}} := \bigcup_{F \in \mathfrak{E}_{\text{ea}}} F$ be the union over all this event sets. At most one of the alternatives may be scheduled.

$$x_j^X + x_{j'}^X \leqslant 1 \quad \text{for all } F \in \mathfrak{E}_{\text{ea}}, j \in F, j' \in E_{\text{ea}} \backslash F \tag{4.127}$$

Event location: Let el be a constraint of this type. Let $E_{\text{el}} \subseteq E$ be the set of events and $L_{\text{el}} \subseteq L$ the set of locations that are given by el. None of the events in E_{el} may have a location in $L \backslash L_{\text{el}}$. This constraint is already considered by generating the set J of allowed combinations for event activities. If desired, all scheduled events in E_{el} must have the same location:

$$x_{j,l}^L \leqslant x_{j',l}^L + 2 - x_j^X - x_{j'}^X \quad \text{for all } l \in L_{\text{el}}, j, j' \in E_{\text{el}} \tag{4.128}$$

Allow non-local resources: An assignment $i \in A$ cannot get a resource that has to travel longer than the maximum allowed travel time t_i^{tr}.

$$\sum_{l,l' \in L_{(i,r)}} \lambda_{r,t,l,l'} \cdot \tau_{l,l'} \leqslant t_{\max}^{\text{tr}} + (t_i^{\text{tr}} - t_{\max}^{\text{tr}}) \cdot \sum_{l \in L_{(i,r)}} y_{i,s(t,i,r),l,r}$$
$$\text{for all } i \in A, r \in R_{(i)}, t \in T_{(i,r)} \tag{4.129}$$

Herein, $\tau_{l,l'}$ is the required travel time from location l to location l', $t_{\max}^{\text{tr}}$ is the maximum possible travel time between two locations and $s(i,t,r)$ is the earliest possible time of activity for assignment i with resource r after t.

Associated resources: Let $a = a_{\text{ara}}^{\text{master}} \in A$ be the master assignment and $A_{\text{ara}}^{\text{client}} \subseteq A$ be the set of client assignments that belong to a constraint ara of this type. For $r \in R$, we denote by $R_{\text{ara},r}$ the set of associated resources of r. We further use variables ara_r to indicate if r has associated resources ($\text{ara}_r = 1$) or if R_r^{ara} is empty ($\text{ara}_r = 0$). If the use of associated resources is required, we must ensure the following:

- Every scheduled client assignment must use a resource which has an empty set of associated resources.
- If any client assignment is scheduled, the master assignment must be scheduled, too.
- The resources of all scheduled client assignments must be associated resources of the master assignments resource.
- If the master assignment is scheduled and uses resource $r \in R$, there must exist a scheduled client assignment using one of the associated resources of r.

This is done by constraints (4.130)-(4.133).

$$\sum_{r \in R} y_{i,r}^R (1 - \text{ara}_r) \geqslant y_i^X \quad \text{for all } i \in A_{\text{ara}}^{\text{client}} \tag{4.130}$$

$$y_a^X \geqslant y_i^X \quad \text{for all } i \in A_{\text{ara}}^{\text{client}} \tag{4.131}$$

$$\sum_{r' \in R_{\text{ara},r}} y_{i,r'}^R \geqslant y_{a,r}^R \quad \text{for all } i \in A_{\text{ara}}^{\text{client}}, r \in R \tag{4.132}$$

$$\sum_{i \in A_{\text{ara}}^{\text{client}}} \sum_{r' \in R_r^{\text{ara}}} y_{i,r'}^R \geqslant y_{a,r}^R \quad \text{for all } r \in R \tag{4.133}$$

If the use of associated resources is optional, only (4.130)-(4.132) have to be satisfied.

Count filter in periods: Let cf be a rule of this type and R_{cf}, A_{cf} sets of resources and filter assignments that are defined by cf. For all $r \in R_{\text{cf}}$ the times of activity or inactivity are measured w.r.t. the assignments in A_{cf} and rolling periods. Let $M_{\text{cf},1}, \cdots, M_{\text{cf},k_{\text{cf}}}$ be the system of time sets that have to be checked. Here, we consider the case that the shortest consecutive time of activity/inactivity must be greater than a given bound $\text{min}_{\text{cf}} > 1$.

Times must either be measured in terms of minutes (converted to time grid units) or in terms of resource days. We set

$$\widetilde{z}^c := \begin{cases} z^{\text{cons}}, & \text{if unit is minutes} \\ z^{\text{consD}}, & \text{else} \end{cases}$$

and can model the rule by

$$1 \leqslant \sum_{t \in M_{\text{cf},k}} \widetilde{z}^c_{\text{cf},t,r} \quad \text{for all } r \in R_{\text{cf}}, k \in [k_{\text{cf}}] \tag{4.134}$$

Recall, that the value $\min_{\text{cf}}$ is incorporated in $\widetilde{z}^c$, which indicates if a certain point in time belongs to an interval of consecutive activity as required.

No resource sharing across locations: The fact that every resource has exactly one location at a time is stated by (4.93).

Assignment distance: A constraint ad of this type defines two sets $A_{\text{ad},1}$ and $A_{\text{ad},2}$ of assignments that must observe given limits $\min_{\text{ad}}$ and/or $\max_{\text{ad}}$ w.r.t. their distance in time. The distance is either measured in terms of minutes (converted to time grid units) or in terms of resource days. The constraint ad further defines one of three possible modes of distance. In the first mode, the limitations concern the gap between the bounding periods of both assignment lists. This can be written by (4.135a)(4.136a) (unit minute) or (4.135b)(4.136b) (unit resource day).

$$(\sum_{t \in T} y^{\text{GS}}_{\text{ad},t}) \cdot \min_{\text{ad}} \leqslant \text{GE}_{\text{ad}} - \text{GS}_{\text{ad}} \tag{4.135a}$$

$$(\sum_{t \in T} y^{\text{GS}}_{\text{ad},t}) \cdot \min_{\text{ad}} \leqslant \text{GED}_{\text{ad}} - \text{GSD}_{\text{ad}} \tag{4.135b}$$

$$\text{GE}_{\text{ad}} - \text{GS}_{\text{ad}} \leqslant \max_{\text{ad}} \tag{4.136a}$$

$$\text{GSD}_{\text{ad}} - \text{GED}_{\text{ad}} \leqslant \max_{\text{ad}} \tag{4.136b}$$

Additionally, in the second mode, for every assignment $i \in A_{\text{ad},1} \cup A_{\text{ad},2}$ that starts directly after the gap, no time t that is allowed for i by a time frame rule must be contained in the interval $U = [\text{GS}_{\text{ad}} + \text{plus}, \text{GE}_{\text{ad}} - \text{minus}]$:

$$y^{AS}_{i,t+1} + y^{\text{GE}}_{\text{ad},t} + \frac{1}{N} \sum_{s \in T_{(i,l,r)}} y^{U}_{\text{ad},s} \leqslant 2$$

$$\text{for all } i \in A_{\text{ad},1} \cup A_{\text{ad},2}, l \in L_{(i)}, r \in R_{(i,l)}, t \in T \tag{4.137}$$

In the third mode, the limits concern the distance from the start of the first assignment list to the start of the second assignment list and may also be negative:

$$\min_{\text{ad}} \leqslant \text{LS}_{\text{ad},2} - \text{LS}_{\text{ad},1} \leqslant \max_{\text{ad}} \tag{4.138}$$

Travel time: For each two locations $l, l'i \in L$ let $\tau_{l,l'}$ be the time in time grid units that is required to travel from l to l'. If a resource has location l at time $t \in T$ and location l' at time $t+1$, we may assume that it is duty at time $t+1$. Under this assumption, it must not be duty for the travel time duration before $t+1$:

$$\lambda_{r,t,l,l'} + \frac{1}{\tau_{l,l'}} \sum_{s=t+1-\tau_{l,l'}}^{t} z_{r,s} \leqslant 1 \quad \text{for all } r \in R, t \in T_{(r)}, l, l' \in L \tag{4.139}$$

Chapter 5

Two-Color Discrepancy of Arithmetic Progressions

5.1 Introduction

5.1.1 The Problem

Two-Color Discrepancy

Let $\mathcal{H} = (V, \mathcal{E})$ denote a finite *hypergraph*, where V is a finite set (called *vertices* or *nodes*) and $\mathcal{E}$ is a family of subsets of V (called *hyperedges*). Let $n = |V|$ and $m = |\mathcal{E}|$. A two-coloring of $\mathcal{H}$ is a function $\chi : V \to C$, where C is a set of cardinality 2. For convenience, we choose $C = \{-1, +1\}$ and may think of -1 as color "red" and 1 as color "blue". The color classes $\chi^{-1}(-1)$ and $\chi^{-1}(+1)$ are a partition of V. Let $E \in \mathcal{E}$ be a hyperedge. With $\chi(E) := \sum_{i \in E} \chi(i)$, the number $|\chi(E)|$ is the imbalance of the two colors in E and we define the discrepancy of $\mathcal{H}$ with respect to the coloring χ by

$$\operatorname{disc}(\mathcal{H}, \chi) = \max_{E \in \mathcal{E}} |\chi(E)|. \tag{5.1}$$

Let $\mathcal{C}_2(\mathcal{H})$ be the set of all two-colorings of $\mathcal{H}$. The hypergraph parameter

$$\operatorname{disc}(\mathcal{H}) = \min_{\chi \in \mathcal{C}_2(\mathcal{H})} \operatorname{disc}(\mathcal{H}, \chi) \tag{5.2}$$

is called the (combinatorial) discrepancy of $\mathcal{H}$.

Arithmetic Progressions

For $a, d, l \in \mathbb{N}$ denote by $A_{adl} := \{a + id \,|\, 0 \leqslant i \leqslant l - 1\}$ the arithmetic progression with starting point a, difference d and length l. Denote by

$\mathcal{E}_n$ the set of all arithmetic progressions in the first n positive integers $[n] = \{1, \cdots, n\}$, that is $\mathcal{E}_n = \{A_{adl} \cap [n] | a, d, l \in [n]\}$. Set $\mathcal{A}_n = ([n], \mathcal{E}_n)$. $\mathcal{A}_n$ is the hypergraph of arithmetic progressions in the first n positive integers.

5.1.2 Related Work

A celebrated result of Spencer [Spe85], the six-standard-deviation theorem, says that for a hypergraph $\mathcal{H}$ with $n = m$, we have $\mathrm{disc}(\mathcal{H}) \leqslant 6\sqrt{n}$. Since hypergraphs with a Hadamard matrix of order n as their incidence matrix have discrepancy at least $c\sqrt{n}$, for a constant $c > 0$, the upper bound is tight up to a constant. Spencer's proof that a two-coloring with discrepancy at most $6\sqrt{n}$ does exist, relies on the pigeonhole principle applied to the exponential large space $\mathcal{C}_2(\mathcal{H})$. It has been a long-standing open problem whether or not there is a polynomial-time algorithm which can construct an $\mathcal{O}(\sqrt{n})$-discrepancy two-coloring. It is known that already the simple randomized algorithm where we choose a random two-coloring of $\mathcal{H}$ by flipping a fair coin, independently for each vertex, returns for any $0 < p < 1$ with probability at least p a two coloring with discrepancy at most $\mathcal{O}(\sqrt{n \log(\frac{m}{p})})$. Moreover, this algorithm can be derandomized, which means it can be converted to a polynomial-time algorithm retrieving a two-coloring with the same asymptotic discrepancy bound. So for $n = m$ the algorithmically reachable discrepancy bound of this algorithm is of the asymptotic order $\sqrt{n \log(n)}$. A positive answer to the above question was recently given by Bansal and Spencer, who provide a deterministic polynomial-time two-coloring algorithm with asymptotic order $\sqrt{n}$ [BS11].

Beside its difficulty, two-color hypergraph discrepancy is a fundamental combinatorial problem with far-reaching applications, i. e., to the design of efficient algorithms in e. g., combinatorial optimization [MNN94] [Sri01a] [KS08], in computational geometry [Mat00] and numerical integration. Combinatorial discrepancy theory is a classical but vivid area in combinatorics [BC87] [BS95] [Mat99] [Cha00] [AS08].

For the hypergraph $\mathcal{A}_n$ of arithmetic progressions in $[n]$ the gap between the optimal discrepancy bound and what is efficiently computable is even an order of magnitude. Roth [Rot64] proved the celebrated lower bound $\mathrm{disc}(\mathcal{A}_n) = \Omega(\sqrt[4]{n})$. Roth himself believed that this bound was too small and that the discrepancy actually should be close to $\sqrt{n}$. This was disproved by Sárközy (see [ES74]), who showed an upper bound of $\mathcal{O}(\sqrt[3]{n \log n})$. Inventing the partial coloring method, Beck [Bec81] showed a nearly tight bound of $\mathcal{O}(\sqrt[4]{n}\sqrt[5]{(\log n)^2})$. Finally Matoušek and Spencer [MS96] solved the discrepancy problem for $\mathcal{A}_n$ by proving the asymptotically tight upper bound $\mathcal{O}(\sqrt[4]{n})$. From these results we now know that the minimum discrepancy for the graph of arithmetic progressions is of the asymptotic order $\sqrt[4]{n}$. The probabilistic method gives a randomized polynomial-time algorithm returning a two-coloring for $\mathcal{A}_n$ of discrepancy at most $\mathcal{O}(\sqrt{n \log(n)})$, while the polynomial-time algorithm of Sárközy stated in [ES74] can compute a $\mathcal{O}(\sqrt[3]{n \log n})$ discrepancy coloring. But the algorithmically fundamental problem of computing a two-coloring of discrepancy $\mathcal{O}(\sqrt[4]{n})$ remains unsolved. Since we know what the asymptotic order of the minimum discrepancy for $\mathcal{A}_n$ is, and since no efficient algorithm for computing a minimum discrepancy two-coloring of $\mathcal{A}_n$ is known, it can serve as a challenging benchmark problem for the progress of algorithmic advances in the area of discrepancy theory.

5.1.3 Complexity of the Two-Color Discrepancy Problem

A zero discrepancy coloring for a simple graph, where each edge consists of exactly 2 vertices, is nothing else than a two-coloring in the well-known sense of graph theory. Whether or not there is a zero-discrepancy two-coloring of graphs is decidable in polynomial time, as it is equivalent to the bipartiteness of the graph. In contrast to this fact, for hypergraphs the same problem is $\mathcal{NP}$-complete [Kni97]. Thus, finding low discrepancy two-colorings of a hypergraph is a complexity-theoretic hard problem.

5.2 Theoretical Results

There are constants c and C such that $c\sqrt[4]{n} \leqslant \operatorname{disc}(\mathcal{A}_n) \leqslant C\sqrt[4]{n}$ holds for all $n \in \mathbb{N}$. While an explicit value for the constant C of Matoušek and Spencer is not known, Roth showed that $1/\pi$ is a suitable choice for c. A crucial point in Roth's proof is the lower bound $1/\pi^2$ for the following function

$$g(n,\alpha) = \frac{1}{n} \sum_{j=1}^{\lfloor\sqrt{n}\rfloor} \left| \sum_{k=0}^{\lfloor\frac{\sqrt{n}}{2}\rfloor-1} e^{2\pi ijk\alpha} \right|^2, \quad n \in \mathbb{N}, \alpha \in [0,1].$$

Numerical calculations show that, already for small n, this bound is quite tight. For example, we have that $\min_\alpha g(n,\alpha)$ approximatively is 0.16 for $n = 1000$ and 0.148 for $n = 10000$.

5.2.1 The Size of $\mathcal{A}_n$

The computational effort of our two-coloring algorithms depends on the number $|\mathcal{E}_n|$ of hyperedges of $\mathcal{A}_n$. For that reason we give an estimation of this number, here

5.1 Proposition. *The number of hyperedges in $\mathcal{A}_n$ is*

$$|\mathcal{E}_n| = n + \sum_{a=1}^{n-1} \sum_{d=1}^{n-a} \lfloor \frac{n-a}{d} \rfloor.$$

which is asymptotically close to $\frac{n^2\log(n)}{2}$, i. e., $\Theta(n^2\log(n))$.

The number of single vertex edges in $\mathcal{A}_n$ is n and for given $a \leqslant n-1$ and $d \leqslant n-a$, the term $\lfloor\frac{n-a}{d}\rfloor$ counts the exact number of arithmetic progressions with starting point a, difference d and length greater than 1. This proofs the exact formula. In order to proof the asymptotic estimation of the exact number of hyperedges, we use the following facts

5.2 Lemma. *Let $n \in \mathbb{N}$, $H_n := \sum_{i=1}^n \frac{1}{i}$ be the n-th partial sum of the harmonic series and $S_n := \sum_{i=1}^n i\log(i)$. It holds that*

$$\log(n) \leqslant H_n \leqslant \log(n) + 1, \tag{5.3}$$

$$\frac{n^2\log(n)}{2} - \frac{n^2}{4} + \frac{1}{4} \leqslant S_n \leqslant \frac{(n+1)^2\log(n+1)}{2} - \frac{(n+1)^2}{4} + \frac{1}{4}. \tag{5.4}$$

Inequalities (5.3) are well known. For the proof of (5.4), we also use the fact that the range of sums of the form $\sum_{i=1}^n f(i)$ with monotonically decreasing/increasing function $f : \mathbb{R} \to \mathbb{R}$ can be estimated by accordant integrals:

5.3 Lemma. *Let $a, b \in \mathbb{N}$ and $f : [a, b+1] \to \mathbb{R}$ be monotonically decreasing and $g : [a, b+1] \to \mathbb{R}$ monotonically increasing. It holds that*

$$\int_{t=a}^{b+1} f(t)dt \leqslant \sum_{i=a}^{b} f(i) \leqslant f(a) + \int_{t=a}^{b} f(t)dt, \tag{5.5}$$

$$g(a) + \int_{t=a}^{b} g(t)dt \leqslant \sum_{i=a}^{b} g(i) \leqslant \int_{t=a}^{b+1} g(t)dt. \tag{5.6}$$

Proof. Define $\overline{f} : [a, b+1] \to \mathbb{R}$ to be the piecewise constant function satisfying

$$\forall i \in \mathbb{N} \cap [a, b]\ \forall t \in [i, i+1) : \overline{f}(t) = f(i).$$

Then, $f(t) \leqslant \overline{f}(t)$ for all $t \in [a, b+1]$ and $\overline{f}(t) \leqslant f(t+1)$ for all $t \in [a, b]$. We further have

$$\sum_{i=a}^{b} f(i) = \int_{t=a}^{b+1} \overline{f}(t)dt.$$

This proofs (5.5), while (5.6) corresponds to (5.5) with $f = -g$. $\square$

Proof of Lemma 5.2 (5.4). The function $F : \mathbb{R}_{\geqslant 1} \to \mathbb{R}, t \mapsto \frac{t^2 \log(t)}{2} - \frac{t^2}{4}$ satisfies $F'(t) = t \log(t)$ for all $t \in \mathbb{R}_{\geqslant 1}$. This implies

$$\frac{n^2 \log(n)}{2} - \frac{n^2}{4} + \frac{1}{4} = \int_{t=1}^{n} t \log(t)dt = \int_{t=2}^{n+1} (t-1)\log(t-1)dt$$

which, due to Lemma 5.3 (5.6), is bounded by

$$\sum_{i=2}^{n} i \log(i) = S_n.$$

The second inequality of (5.4) is also implied by Lemma 5.3 (5.6):

$$S_n \leqslant \int_{t=1}^{n+1} t \log(t)dt = \frac{(n+1)^2 \log(n+1)}{2} - \frac{(n+1)^2}{4} + \frac{1}{4}.$$

$\square$

Now, for an $a \leqslant n-1$, (5.3) gives

$$\sum_{d=1}^{n-a} \lfloor \frac{n-a}{d} \rfloor \leqslant \sum_{d=1}^{n-a} \frac{n-a}{d} \leqslant (n-a)\log(n-a) + (n-a), \tag{5.7}$$

$$\sum_{d=1}^{n-a} \lfloor \frac{n-a}{d} \rfloor \geqslant \sum_{d=1}^{n-a} (\frac{n-a}{d} - 1) \geqslant (n-a)\log(n-a) - (n-a). \tag{5.8}$$

Inserting (5.7)(5.8) into the exact formula for $|\mathcal{E}_n|$ and using (5.4) we obtain

$$\begin{aligned} |\mathcal{E}_n| &\geqslant n + \sum_{a=1}^{n-1}((n-a)\log(n-a) - (n-a)) = n + \sum_{a=1}^{n-1}(a\log(a) - a) \\ &\geqslant n + \frac{(n-1)^2\log(n-1)}{2} - \frac{(n-1)^2}{4} + \frac{1}{4} - \frac{n(n-1)}{2}, \end{aligned}$$

$$\begin{aligned} |\mathcal{E}_n| &\leqslant n + \sum_{a=1}^{n-1}((n-a)\log(n-a) + (n-a)) = n + \sum_{a=1}^{n-1}(a\log(a) + a) \\ &\leqslant n + \frac{n^2\log(n)}{2} - \frac{n^2}{4} + \frac{1}{4} + \frac{n(n-1)}{2}, \end{aligned}$$

which proofs the claim that $\mathcal{E}_n$ is $\Theta(n^2\log(n))$.

5.3 Exact Algorithm

5.3.1 Limits of Integer Linear Programming

The two-coloring problem of $\mathcal{A}_n$ has an equivalent formulation as integer linear program (ILP):

$$\begin{aligned} \min \quad & disc \\ s.t. \quad & 2\sum_{v \in E} x_v + disc \geqslant |E| \quad \forall E \in \mathcal{E}_n, \\ & 2\sum_{v \in E} x_v - disc \leqslant |E| \quad \forall E \in \mathcal{E}_n, \\ & x \in \{0,1\}^n, \\ & disc \in \mathbb{N}. \end{aligned}$$

The relationship between a solution vector $x \in \{0,1\}^n$ of the ILP and a coloring of the form $\chi : V \to \{-1,1\}$ is given by $\chi(v) = 2x_v - 1$.

Standard solvers like CPLEX can be applied to this formulation and provide optimal two-colorings for very small n and upper bounds for larger n. But the application of standard solvers is limited for memory reasons, since the problem matrix of the ILP formulation isn't very sparse, has n columns and about $n^2 \log(n)$ rows (two rows for every hyper edge). We applied CPLEX with default settings to the ILP formulation above. The discrepancy of $\mathcal{A}_n$ is 2 for $n \in \{3,\dots,8\}$, 3 for $n \in \{9,\dots,26\}$ and 4 for $n \in \{27,\dots,60\}$. For the first instance with discrepancy 5, $\mathcal{A}_{61}$, CPLEX required about 40 minutes running time to solve the corresponding ILP.

5.4 Heuristics

5.4.1 Random Half-Half Coloring

A simple random strategy of 2-coloring a hypergraph's vertices is the random two-coloring. We independently flip a fair coin for each vertex of the hypergraph to decide its color. For this algorithm an upper discrepancy bound of $\mathcal{O}(\sqrt{n \log n})$ is easily derived by the probabilistic method [AS08].

5.4.2 Algorithm of Sárközy

The algorithm of Sárközy stated in [ES74] runs in polynomial-time. The algorithm first determines a prime number p such that $n \approx \sqrt{p^3 \log p}$ and colors each number $j \in [p-1]$ with the Legendre symbol

$$\left(\frac{j}{p}\right) := \begin{cases} +1, & \text{if } j \text{ is quadratic remainder modulo } p \\ -1, & \text{else} \end{cases},$$

p with one and each number $j = p+1,\dots,n$ with the same color as $j-p$. The following theorem is stated in [ES74].

5.4 Theorem. *The discrepancy of a two-coloring due to Sárközy is $\mathcal{O}(\sqrt[3]{n \log n})$.*

The algorithm of Sárközy has been now for about 35 years the algorithm with the best theoretically worst-case performance for computing the discrepancy of arithmetic progressions, but practical experiments were not documented, yet.

Our first experiments with Sárközy colorings showed some discrepancy variation in dependence of the chosen prime module. Additional to the usage of the Legendre symbol, we also tried to color the first p numbers randomly but balanced w.r.t. the number of red and blue vertices, respectively, while retaining Sárközy's color replication idea for the remaining numbers in $[n]$. For the better prime modules, we observed that between 1 and 5 percent of the randomly generated two-colorings were at least as good as the colorings based on the usage of the Legendre symbol. Since we will deal with such Sárközy based random colorings within our EA frameworks in Section 5.5 and since a proof of Theorem 5.4 seems to have never been published, we proof a corresponding probabilistic result below.

We call a two-coloring χ of $\mathcal{H}$ balanced, if $|\chi(V)| \leqslant 1$, i. e., if its color classes have either the same cardinality (if n is even) or differ by 1 (if n is not even). By the probabilistic method [AS08] one easily obtains a general upper 2-color discrepancy bound of $\sqrt{2n\log(2m)}$ for hypergraphs. A modification concerning balanced two-colorings is given by

5.5 Proposition. *There is a balanced two-coloring χ of $\mathcal{H}$ with*

$$\operatorname{disc}(\mathcal{H},\chi) \leqslant \sqrt{4n\log(4m)} + 1$$

Proof. We first suppose that n is even and use the following random two-coloring procedure, called RandomBalance(n). Let $V_1, V_2 \subseteq V$ with $|V_1| = |V_2| = n/2$ be a partition of V and φ be a bijection between V_1 and V_2. For $v \in V_1$ let X_v be independent random variables with

$$\mathbb{P}[X_v = +1] = \mathbb{P}[X_v = -1] = \frac{1}{2}.$$

RandomBalance(n) generates a balanced random two-coloring χ of $\mathcal{H}$ by setting $\chi(v) = X_v$ and $\chi(\varphi(v)) = -X_v$ for all $v \in V_1$. Now, the proof is similar to that of [AS08] for general two-colorings. We set $\alpha = \sqrt{n\log(4m)}$. For every hyperedge $E \in \mathcal{E}$, it holds that

$$\begin{aligned}\mathbb{P}\left[|\chi(E)| > 2\alpha\right] &\leqslant \mathbb{P}\left[|\chi(E\cap V_1)| > \alpha \ \vee \ |\chi(E\cap V_2)| > \alpha\right] \qquad (5.9)\\ &\leqslant \mathbb{P}\left[|\chi(E\cap V_1)| > \alpha\right] + \mathbb{P}\left[|\chi(E\cap V_2)| > \alpha\right].\end{aligned}$$

For $i \in \{1, 2\}$, application of Chernoff bounds gives

$$\begin{aligned}\mathbb{P}\left[|\chi(E \cap V_i)| > \alpha\right] &= \mathbb{P}\left[|\sum_{v \in E \cap V_i} X_v| > \alpha\right] \\ &< 2e^{-\alpha^2/(2|E \cap V_i|)} \leqslant 2e^{-\alpha^2/n} = \frac{1}{2m}.\end{aligned} \tag{5.10}$$

By (5.9) and (5.10) we have $\mathbb{P}\left[|\chi(E)| > 2\alpha\right] < \frac{1}{m}$, which yields

$$\mathbb{P}\left[\exists E \in \mathcal{E} :\ |\chi(E)| > 2\alpha\right] \leqslant \sum_{E \in \mathcal{E}} \mathbb{P}\left[|\chi(E)| > 2\alpha\right] < 1.$$

Thus, there exists a balanced two-coloring χ of $\mathcal{H}$ such that $|\chi(E)| \leqslant 2\alpha$ holds for every hyperedge $E \in \mathcal{E}$. In the case that n is not even, we may pick one vertex $v \in V$ and consider the subhypergraph of $\mathcal{H}$ that is induced by $V \backslash \{v\}$. Its discrepancy is also bounded by $\sqrt{4n \log(4m)}$ and the discrepancy of $\mathcal{H}$ is at most one more. $\square$

5.6 Remark. We can easily modify the existence statement of Proposition 5.5 into a probability dependent statement by introducing an $\varepsilon > 0$. Then, with probability greater than $1 - \varepsilon$ the two-coloring generated by RandomBalance(n) has discrepancy at most $\sqrt{4n \log\left(\frac{4m}{\varepsilon}\right)} + 1$.

In the following, we utilize arithmetic progressions in $\mathbb{Z}_p$. An arithmetic progression in $\mathbb{Z}_p$, p prime, consists of all residual classes that are represented by the elements of some arithmetic progression $A_{a,d,l}$. Denote by $\mathcal{E}_{\mathbb{Z}_p}$ the set of all arithmetic progressions in $\mathbb{Z}_p$. Set $\mathcal{A}_{\mathbb{Z}_p} = (\mathbb{Z}_p, \mathcal{E}_{\mathbb{Z}_p})$. $\mathcal{A}_{\mathbb{Z}_p}$ is the hypergraph of arithmetic progressions in $\mathbb{Z}_p$. The number of hyperedges in $\mathcal{A}_{\mathbb{Z}_p}$ is bounded by p^3, since $\mathcal{E}_{\mathbb{Z}_p}$ is induced by arithmetic progressions $A_{a,d,l}$ with $a \in \{0, \ldots, p-1\}$ and $d, l \in [p]$. For this reason, we obtain

5.7 Corollary. *Let p be a prime number and $\varepsilon > 0$. With probability greater than $1 - \varepsilon$ RandomBalance(p) finds a two-coloring $\chi_{\mathbb{Z}_p}$ of $\mathcal{A}_{\mathbb{Z}_p}$ with*

$$\operatorname{disc}(\mathcal{A}_{\mathbb{Z}_p}, \chi_{\mathbb{Z}_p}) \leqslant \sqrt{4p \log(\frac{4p^3}{\varepsilon})}.$$

Choosing $x \in \mathbb{R}_{>0}$ such that $n = \sqrt{x^3 \log x}$, the assumption $n \approx \sqrt{p^3 \log p}$ about the prime module p for the algorithm of Sárközy can be specified to $p \in [\frac{x}{2}, x]$, which exists due to Bertrands postulate and yields

$\sqrt{p^3 \log p} \in [\frac{n}{4}, n]$. Now, using Corollary 5.7, we can proof the following version of Theorem 5.4.

5.8 Theorem. *Let p be a prime module such that $\sqrt{p^3 \log p} \in [\frac{n}{4}, n]$. Let χ be a two-coloring of $\mathcal{A}_{[n]}$, where $\chi|_{[p]}$ is obtained by RandomBalance(p) and $\chi(v) := \chi(v-p)$ for $v \in [n]\backslash[p]$. We have* $\operatorname{disc}(\mathcal{A}_{[n]}, \chi) \geqslant \sqrt[3]{\frac{1}{9} n \log n}$ *and with probability greater than $\frac{1}{2}$ we have* $\operatorname{disc}(\mathcal{A}_{[n]}, \chi) \leqslant 8\sqrt[3]{\frac{2}{3} n \log n}$.

Proof. The arithmetic progression $A_{1,p,\lceil \frac{n}{p} \rceil}$ is monochromatic w.r.t. χ and thus has discrepancy at least

$$\frac{n}{p} \geqslant \sqrt{p \log p}.$$

It holds that

$$\log \frac{n}{4} \leqslant \log \sqrt{p^3 \log p} \leqslant \log(p^2) = 2 \log p,$$

which implies $\log n \leqslant \frac{9}{4} \log p$ and yields

$$9\sqrt{p^3 (\log p)^3} = 9 \log(p) \sqrt{p^3 \log p} \geqslant n \log n,$$

i. e.,

$$\operatorname{disc}(\mathcal{A}_{[n]}, \chi) \geqslant \frac{n}{p} \geqslant \sqrt{p \log p} \geqslant \sqrt[3]{\frac{1}{9} n \log n}.$$

The prime module p further satisfies

$$\frac{n}{p} \leqslant 4\sqrt{p \log p} \tag{5.11}$$

and

$$\frac{3}{2}\sqrt{p^3 (\log p)^3} = \frac{3 \log p}{2} \sqrt{p^3 \log p} = \log\left(\sqrt{p^3}\right) \sqrt{p^3 \log p} \leqslant n \log n,$$

i. e.,

$$\sqrt{p \log p} \leqslant \sqrt[3]{\frac{2}{3} n \log n}. \tag{5.12}$$

Let $A_{a,d,l}$ be any arithmetic progression in $[n]$ with starting point a, difference d and length l. Due to (5.11) and (5.12) it suffices to show that $|\chi(A_{a,d,l})| \leqslant \frac{n}{p} + 4\sqrt{p \log p}$ with probability greater than $\frac{1}{2}$. We distinguish the cases $\gcd(d,p) = p$ and $\gcd(d,p) = 1$. In the first case, d is multiple of p which yields $|\chi(A_{a,d,l})| \leqslant l \leqslant \frac{n}{p}$. In the second case, let $l = qp + r$ with $q \in \mathbb{N}_0$, $q \leqslant \frac{n}{p}$ and $r \in \{0,\dots,p-1\}$. The arithmetic progression $A_{a,d,l}$ has a partition into q arithmetic progressions of length p and one arithmetic progression of length r:

$$A_{a,d,l} = \bigcup_{k=0}^{q-1} A_{a+kpd,d,p} \cup A_{a+qpd,d,r}$$

Since d and p are coprime, the q subprogressions of length p correspond to the full arithmetic progression in $\mathbb{Z}_p$. Thus, by the construction of χ all these subprogressions have discrepancy 1 (w.r.t. χ). The last subprogression $A_{a+qpd,d,r}$ of $A_{a,d,l}$ corresponds to some arithmetic progression in $\mathbb{Z}_p$ which implies that its discrepancy is at most $\operatorname{disc}(\mathcal{A}_{\mathbb{Z}_p}, \chi|_{[p]})$. After Corollary 5.7 with probability greater than $\frac{1}{2}$ the latter is bounded by

$$\sqrt{4p\log(8p^3)} = \sqrt{4p(\log(8) + 3\log(p))} \leqslant \sqrt{16p\log(p)} = 4\sqrt{p\log p}.$$

Thus we have $|\chi(A_{a,d,l})| \leqslant \frac{n}{p} + 4\sqrt{p\log p}$ with probability greater than $\frac{1}{2}$ as desired. □

5.4.3 Local Search

A given two-coloring may possibly be improvable by changing the color of some single vertex or by swapping the colors of two differently colored vertices, respectively. We can test all vertices, resp. pairs of vertices, and apply the corresponding variation. The procedure can be repeated until no improvement is possible any more. We implemented this local search procedure (LS) as a further benchmark heuristic. In each iteration, the faster bit-flip step is tried first while, optionally, the bit-swap step is only tried if bit-flip is not successful. We denote the variant that only tries to flip bits by LS_1 and the variant that also tries to swap bits by LS_2.

5.5 EA Framework

Since the two-coloring problem is of purely binary nature, we apply a classical binary EA framework and the QiEA framework after Han and Kim (see Section 1.5), here.

5.5.1 Genotypes and Encoding

In order to proof the upper discrepancy bound of $\mathcal{O}(\sqrt[3]{n}\log n)$ (cf., Theorem 5.4), Sárközy chose a prime module p such that $n \simeq \sqrt{p^3 \log p}$ and constructed a two-coloring of $[n]$ by repeating a generating Legendre symbol coloring of $[p]$. We may two-color modulo p within other algorithms, e. g., replacing the Legendre symbol with the outcomes of random variables. The Legendre symbol is symmetric in the sense that

$$\left(\frac{p-j}{p}\right) = -\left(\frac{j}{p}\right), \quad j \in [p-1].$$

We further know that an optimal two-coloring of $\mathcal{A}_n$ must be balanced [DS03]. For that reason, our genotypes will be binary strings b of length $\frac{p+1}{2}$. The encoding function that yields the corresponding coloring χ of $[n]$ is defined by

$$\chi(v) := \begin{cases} 2b_v - 1, & \text{if } v \in [\frac{p-1}{2}] \\ 1 - 2b_{p-v}, & \text{if } v \in \{\frac{p+1}{2}, \dots, p-1\} \\ 2b_{\frac{p+1}{2}} - 1, & \text{if } v = p \end{cases} \tag{5.13}$$

for the generating part of χ and by

$$\chi(v) := \begin{cases} \chi(v-p), & \text{if } v \in \{p+1, p+2, \dots, n\} \backslash p\mathbb{Z} \\ \chi(p), & \text{if } v \in \{p+1, p+2, \dots, n\} \cap 2p\mathbb{Z} \\ -\chi p, & \text{if } v \in \{p+1, p+2, \dots, n\} \cap (p\mathbb{Z} \backslash 2p\mathbb{Z}) \end{cases} \tag{5.14}$$

for the remaining vertices. This approach reduces the search space size from 2^n to $2^{\frac{p+1}{2}}$.

We denote with $p(n)$ the smallest prime number that satisfies $n < \sqrt{p(n)^3 \log(p(n))}$. We made the following observations by precalculations.

1. The original Sárközy coloring described in [ES74] is generated by (5.13) and (5.14), if we set $\chi(v) = \chi(v-p)$ in all cases of the second formula. However, the discrepancy results of all examined two-coloring algorithms are better, if we use the balanced version.
2. For a given $n \in \mathbb{N}$, the best prime module for (5.14) using the Legendre

symbol for the generating part of χ, say $p^*(n)$, is also small related to n.

3. For every prime module p we can generate additional random colorings by choosing $b \in \{0,1\}^{\frac{p+1}{2}}$ under uniform distribution and applying (5.13)(5.14). If we repeat the random experiment for many, say 10000, trials and do the same for every prime module between $p(n)$ and n, the discrepancy of the best obtained coloring is often at least as good as that for $p(n)$. Since $p^*(n)$ is sometimes too small to allow for further discrepancy optimization (see initial experiments, Table 5.1) the prime module of our choice, say $p^{**}(n)$, will sometimes differ from $p^*(n)$.

Consequently, the modules $p(n)^{**}$ of 3. are our favorites for the encoding function (5.13)(5.14) within our EA frameworks.

5.5.2 Fitness Function

We use the negative discrepancy as standard fitness function.

Refinement of the Fitness Function

Suppose we have two two-colorings χ and χ', such that $d = \operatorname{disc}(\mathcal{A}_n, \chi) = \operatorname{disc}(\mathcal{A}_n, \chi')$. We can distinguish the qualities of both colorings, if we can measure how far they are from having discrepancy $d-1$. An obvious way to do this is to determine the number m_d of hyperedges with discrepancy d. We can include this number as fractional term into the fitness function by returning $f = -d - \frac{m_d}{m+1}$. We alternatively use this refinement of the fitness function as it should support the ability of an evolutionary algorithm to head for lower discrepancy solutions.

Early Termination of the Fitness Function

During a two-coloring algorithm, whenever the discrepancy d of a current two-coloring is decisive, we may only need to know if it is better than the discrepancy d^* of a former observed two-coloring (e. g., the best one). For this reason it suffices, if our fitness function implementation returns $\max(-d, -d^*)$ instead of $-d$. Since the discrepancy d of a two-coloring χ of $\mathcal{A}_n$ is $\max_{E \in \mathcal{E}_n} |\chi(E)|$ we return $-d^*$, if $|\chi(E)| \geqslant d^*$ holds for some hyperedge $E \in \mathcal{E}_n$. A similar approach is realized for the refined version of the fitness function. Then, we also count the edges that have the same discrepancy as that of the most imbalanced edge w.r.t. the current two-coloring. The fitness function implementation with the options of refinement and early termination is given by Algorithm 7.

Algorithm 7: Genotype Fitness

```
input : Number $n$ of vertices, prime module $p$, $b \in \{0,1\}^{\frac{p+1}{2}}$,
        flag refine, fitness bound $f_{\min}$
output: Fitness value $f$
Generate coloring $\chi$ of $[n]$ from $b$ using (5.13) and (5.14);
$f$ = disc = count = 0;
stop = false;
$a = d = 1$;
while $a \leqslant n$ and not stop do
    while $d \leqslant n + 1 - a$ and not stop do
        $l_{\max} = 1 + \lfloor \frac{n-a}{d} \rfloor$;
        colorsum = 0;
        $l = 1$;
        while $l \leqslant l_{\max}$ and not stop do
            $v = a + (l-1)d$;
            colorsum = colorsum + $\chi(v)$;
            edgedisc = |colorsum|;          // discrepancy of $A_{a,d,l}$
            if edgedisc = disc then count = count + 1;
            if edgedisc > disc then
                disc = edgedisc;
                count = 1;
            $f$ = −disc;
            if refine then $f = f - \frac{\text{count}}{|\mathcal{E}_n|+1}$;
            if $f < f_{\min}$ then
                $f = f_{\min}$;
                stop = true;
            $l = l + 1$;
        $d = d + 1$;
    $a = a + 1$;
```

5.5.3 Operational Settings

Optional EA variation operators are bit-flip mutation, bit-swap mutation, two-point crossover and uniform crossover. The local search heuristics LS_1 and LS_2 from subsection 5.4.3 are further mutation operators. We will try different population (group) sizes for the classical EA where the initial population always consists of randomly generated individuals.

Concerning the QiEA, we use a group size of one, since one individual does already comprise both a good (initial) exploration capacity and a good learning ability. In each generation, we do multiple observations ($N_{\text{obs}} = 5$) of the qubits and replace the associated attractor by the best of all obtained offsprings and the current attractor. Subsequently, we update the probabilities. Rather than applying rotation gates (cf., subsection 1.5.6 and [HK02][HK04]) we change probabilities by simple additive terms $\pm\Delta$ (with overflow prevention) since we observed this to provide better performance. For our settings, we found $\Delta := 10^{-4}$ to be a suitable choice.

Algorithm 8 describes the top level of the classical EA framework for the two-coloring problem. The top level of the QiEA framework is quite similar. Using the simple top level QiEA framework as described above, its pseudo code is quite similar to Algorithm 8. The procedure `iterateEA` (Algorithm 9) will be replaced with `iterateQiEA` (Algorithm 10) and migration is performed by replacing the attractor of the unique individual of each group.

The stopping criterion for all operational parameter settings is a time limit of $n^2 \log(n) * 10^{-5}$ seconds which is proportional to the number of hyper edges.

5.6 Experiments

Similar to our experiments in Section 2.6 and Section 3.5 we use the Linux system with 2100 MHz clock rate AMD Opteron CPUs. We start with the algorithm of Sárközy, using (5.14) and the Legendre symbol for the generating part of the coloring. Table 5.1 shows the discrepancy results and lower discrepancy bounds for both the smallest prime module $p(n)$ that satisfies $n < \sqrt{p(n)^3 \log(p(n))}$ and the best prime module $p^*(n)$ in $[n]$. We further state the same for the prime modules $p^{**}(n)$ that we have chosen considering 10000 additional random two-coloring trials (as described in

Algorithm 8: EA framework for the two-coloring problem

```
input : Number $n$ of vertices, prime module $p$,
        operational parameters
output: Two-coloring $\chi$ of $[n]$
1  Generate initial population $P$ that consists of $N_{\mathrm{gr}}$ groups with $N_{\mathrm{ind}}$
   random binary individuals of length $\frac{p+1}{2}$;
2  $t_{\max} = \frac{n^2 \log n}{10000}$;
3  $t_{\mathrm{mig}} = 0$;
4  while $t_{\mathrm{run}} \leqslant t_{\max}$ do
5      for $j \in [N_{\mathrm{gr}}]$ in parallel do
6          iterateEA($G_j$);
7          take current running time $t_{\mathrm{run}}$;
8          if allowMigration and $t_{\mathrm{run}} - t_{\mathrm{mig}} > \frac{t_{\max}}{10}$ then
9              migrate best individual of all groups to each group by
               replacing the most unfit one in a group;
10             $t_{\mathrm{mig}} = t_{\mathrm{run}}$;
11 obtain $\chi$ from the very best individual using (5.13) and (5.14);
```

subsection 5.5.1). The chosen module $p^{**}(n)$ differs from $p^{*}(n)$ if either a better discrepancy was obtained or the discrepancy associated with $p^{*}(n)$ does not provide enough space for further optimization. We see that the prime modules $p(n)$ are too small, since the corresponding lower discrepancy bounds $\lceil \frac{n}{p} \rceil$ do always exceed the results that are obtained for $p^{*}(n)$ and $p^{**}(n)$.

The principle of dealing with generator bit-strings and obtaining full two-colorings by (5.13)(5.14) will be used for all algorithms under consideration, where the prime numbers $p^{**}(n)$ are the default modules. Moreover, the refined fitness function and the early termination of the fitness function calculation are utilized for all algorithms.

We consider the classical EA framework with a single-group population and make the first guess that

- the prime modules $p^{**}(n)$ determined before,
- a population size of 25 and 5 matings per generation

Algorithm 9: iterateEA

```
input : Group $G$ and operational parameters
1 for $k \in [N_{\mathrm{mate}}]$ do
2     select parent index $i_1 \in [N_{\mathrm{ind}}]$ by linear ranking based RWS (see
      subsection 1.5.3);
3     select parent index $i_2 \in [N_{\mathrm{ind}}]$ by linear ranking based RWS until
      $i_2 \neq i_1$;
4     pick individuals $\mathrm{parent}_1$, $\mathrm{parent}_2$ that correspond to $i_1$, $i_2$;
5     apply the chosen crossover operator to the parents in order to
      generate two children $\mathrm{child}_1$, $\mathrm{child}_2$ or set $\mathrm{child}_1 = \mathrm{parent}_1$ and
      $\mathrm{child}_2 = \mathrm{parent}_2$, if no crossover is desired;
6     apply the chosen mutation operator to both children in order to
      generate two mutants $\mathrm{mutant}_1$, $\mathrm{mutant}_2$ or set $\mathrm{mutant}_1 = \mathrm{child}_1$
      and $\mathrm{mutant}_2 = \mathrm{child}_2$, if no mutation is desired;
7     insert both mutants into group $G$;
8 sort the group members w.r.t. the fitness function and resize the
  group to $N_{\mathrm{ind}}$ by deleting its most unfit members;
```

Algorithm 10: iterateQiEA

```
input : Individual $\mathcal{I} = (p, a)$ and operational parameters
1 for $k \in [N_{\mathrm{obs}}]$ do
2     get binary state $b$ by observation (randomized rounding) of $p$;
3     if the fitness of $b$ is better than the fitness of $a$, replace $a$ with $b$;
4 change all probabilities of $p$ towards the corresponding value of $a$ by
  an amount of $\Delta$;
```

Table 5.1. Initial two-coloring algorithm results (specifications are given in the text).

n	p	d	$\lceil \frac{n}{p} \rceil$	p^*	d^*	$\lceil \frac{n}{p^*} \rceil$	p^{**}	d^{**}	$\lceil \frac{n}{p^{**}} \rceil$
1000	67	16	15	83	15	13	101	13	10
2000	97	23	21	127	19	16	163	17	13
3000	127	24	24	223	22	14	229	21	14
4000	149	30	27	283	24	14	283	24	14
5000	173	32	29	283	26	18	307	25	17
6000	191	34	32	283	27	22	307	27	20
7000	211	34	34	283	29	25	337	28	21
8000	229	35	35	467	30	17	367	28	22
9000	251	39	36	367	32	25	439	30	21
10000	263	40	39	587	32	18	467	30	22
15000	347	44	44	587	37	26	587	35	26
20000	409	49	49	587	41	35	653	40	31
25000	467	54	54	587	45	43	743	43	34
30000	541	56	56	1307	47	23	859	46	35
40000	631	64	64	1307	54	31	1307	54	31
50000	727	69	69	1307	55	39	1307	55	39
60000	821	74	74	1307	59	46	1307	59	46
70000	907	78	78	1307	63	54	1663	64	43
80000	977	82	82	1663	66	49	1663	66	49
90000	1051	89	86	1663	69	55	1663	69	55
100000	1129	89	89	1907	71	53	1907	71	53

- uniform crossover,
- bit-swap mutation,
- the refined fitness function and
- the fitness of the worst individual as a bound for early termination of the fitness calculation

are suitable operational settings. There are too many operational parameter combinations to experiment with all of them. For that reason, we try if a better combination is obtained by varying the six settings from above one

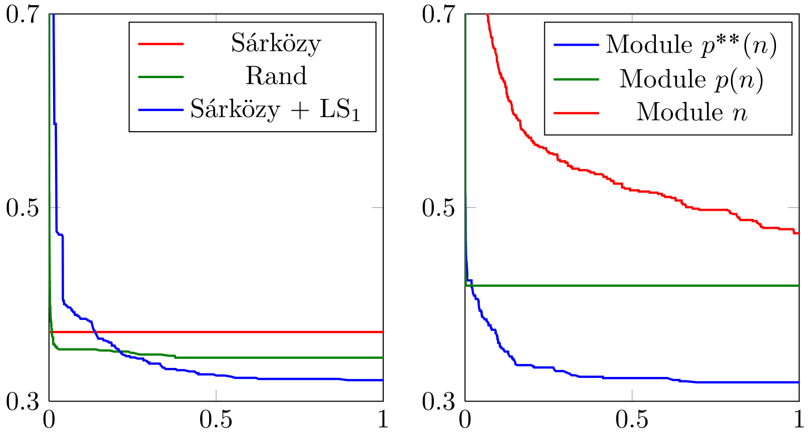


Figure 5.1. Convergence of different heuristics using prime module $p^{**}(n)$ (left) and the classical EA with default settings and different prime modules (right).

by one. We perform our experiments for $n \in \{1000, 2000, \ldots, 10000\}$ and consider normalized convergence plots as we did in Section 2.6. Normalization is done w.r.t. a discrepancy limit of $2\sqrt[3]{n \log n}$ and the runtime limit of $n^2 \log(n) * 10^{-5}$ seconds, which is the stopping criterion for every algorithm. Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4 show the results.

On the left hand side of Figure 5.1 we consider three benchmark heuristics. The best out of repeated random two-coloring trials does soon beat the Legendre symbol generator coloring, but better results are obtained if the generator coloring with the Legendre symbol is improved by local search (see subsection 5.4.3). The right hand side of Figure 5.1 confirms that operating on short binary generator strings which induce the corresponding full colorings yields better results than dealing with strings of full length. Figure 5.2 shows that a population size of 25 with 5 matings per iteration is the best of our three alternatives (w.r.t. the default other operational parameters) and also confirms that uniform crossover behaves well. Since the local search results are already quite good, we decided to incorporate LS as further alternatives for the mutation operator. Figure 5.3 shows that the local search mutation operators, LS$_1$ and LS$_2$, do indeed outperform both,

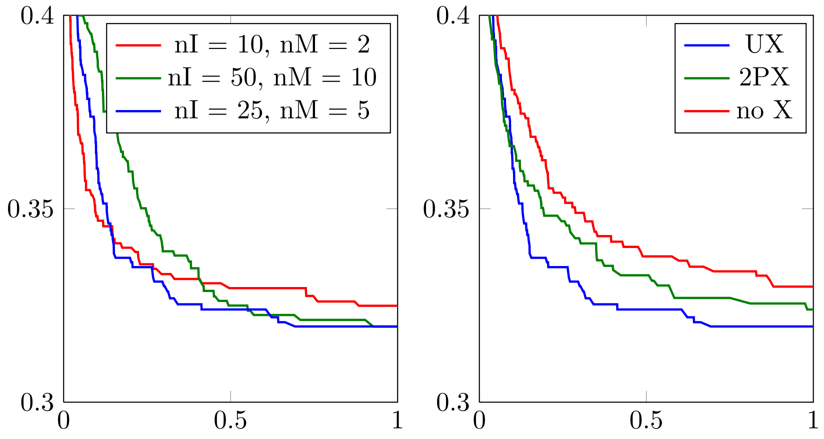


Figure 5.2. Convergence of the classical EA for default settings (blue lines) and other population parameters (left) and crossover operators (right), respectively.

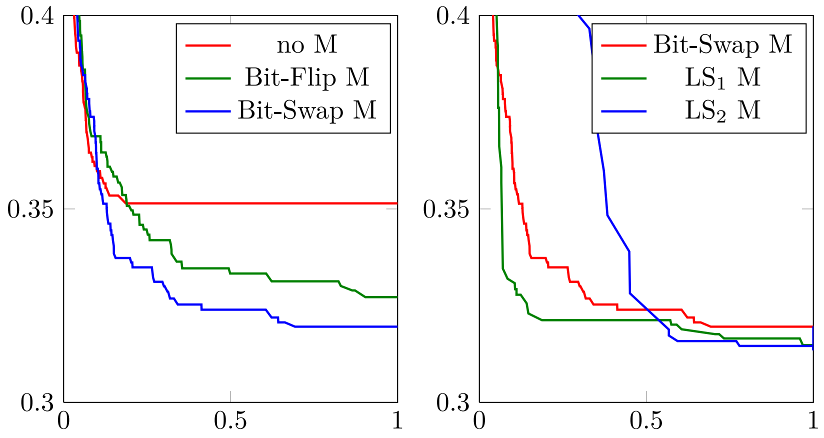


Figure 5.3. Convergence of the classical EA for default settings and other mutation operators.

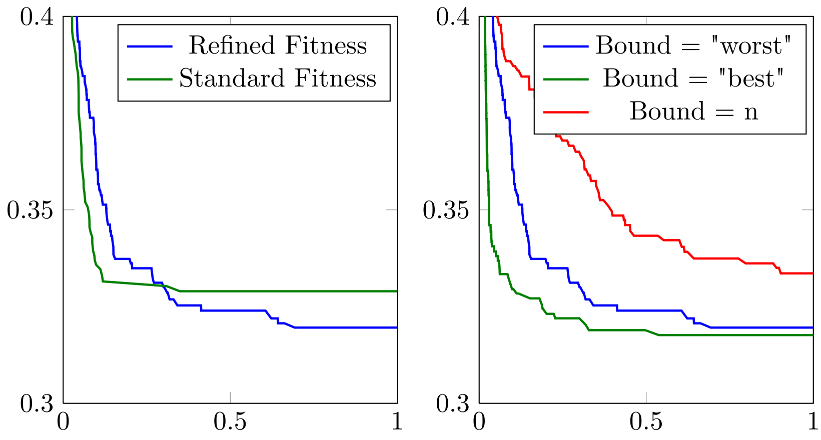


Figure 5.4. Convergence of the classical EA for default settings and alternatives concerning the fitness function.

bit-flip mutation and bit-swap-mutation. From Figure 5.4 we conclude that the refined fitness function yields better results than the standard fitness function. Our conjecture concerning the bound for early termination was that the fitness f_{worst} of the worst individual would be a better choice than the fitness f_{best} of the best individual. The disadvantage using f_{best} is that it is not possible to integrate a new individual into the population the fitness of which lies between f_{best} and f_{worst}. On the other hand, depending on the population size, much more fitness evaluations can be done if f_{best} is used. Referring to Figure 5.4, this advantage seems to be stronger than the disadvantage.

Concerning QiEA we use the framework and settings as described in subsection 1.5.6 and Section 5.5. We refrain from hybridizations with the local search heuristic since we do not want to disturb the QiEA working principle of changing attractors by observation only. Figure 5.5 shows the results for the single-group QiEA compared to the best of our isolated benchmark heuristics as well as to the best found operational setting for the classical single-group EA. In average, QiEA outperforms the benchmark algorithms w.r.t. both efficiency and effectivity.

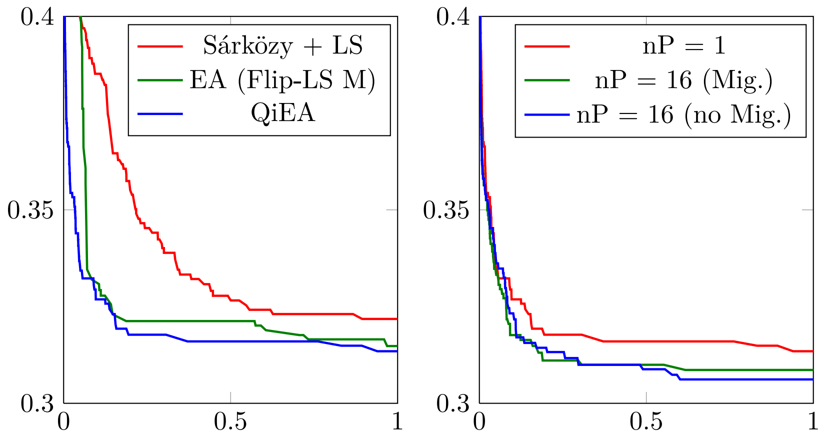


Figure 5.5. Left: Convergence of the best heuristic, the best classical single-group EA setting and the single-group QiEA. Right: Convergence of the QiEA with 1 group, 16 isolated groups and 16 groups with migration.

Finally, we repeat the QiEA experiments with 16 groups instead of one group only, processing each group with its own CPU (i.e. using 16 fold processing power). Similar to the TSP with time windows problem, a somewhat surprising observation is that periodic migration of the very best attractor to all groups does not yield better results than isolated groups. We also observed short migration periods to be inferior to long migration periods. Although qubits are slowly influenced by a migrated attractor, migration seems to cause a loss of diversity which is not compensated by the chance to make more observations in the neighborhood of the good migrant. In the left plot of Figure 5.5 the migration period is 10% of the total running time. Table 5.2 compares the discrepancy results of the modified Sárközy algorithm using the optimal prime modules $p^*(n)$ (also stated in Table 5.1) with the single-group EA (25 individuals, 5 matings per generation, two-point crossover and bit-flip local search mutation), the single-group QiEA and the 16-group QiEA without migration. All EAs ran with the chosen prime modules $p^{**}(n)$. For $n \geqslant 40000$, we only ran the modified Sárközy algorithm and the single-group QiEA since the 16-group QiEA would already

Table 5.2. Two-coloring results of the modified Sárközy algorithm (using prime modules $p^*(n)$) and three EAs (using prime modules p^{**}), respectively.

n	Discrepancies			
	Sárközy	1-group EA	1-group QiEA	16-group QiEA
1000	15	13	13	12
2000	19	17	17	16
3000	22	18	19	18
4000	24	21	21	20
5000	26	22	21	21
6000	27	24	23	23
7000	29	25	24	23
8000	30	25	26	24
9000	32	26	27	25
10000	32	28	28	27
15000	37	32	30	30
20000	41	33	33	33
25000	45	36	35	35
30000	47	37	37	37

require a high quota of the shared computational environment but yield at most little better final discrepancies (cf., $n \leqslant 30000$). The results are shown in Table 5.3 Figure 5.6 shows a plot of the discrepancy results for the 16-group QiEA (blue) in comparison with random two-coloring (green) for $n \in \{1000, 2000, \ldots, 10000\} \cup \{15000, 20000, 25000, 30000\}$ and the chosen prime modules $p^{**}(n)$. It further shows the results of the modified Sárközy algorithm w.r.t. its optimal prime module $p^*(n)$ (red) and the results of random two-coloring applied to the full bit-strings (black).

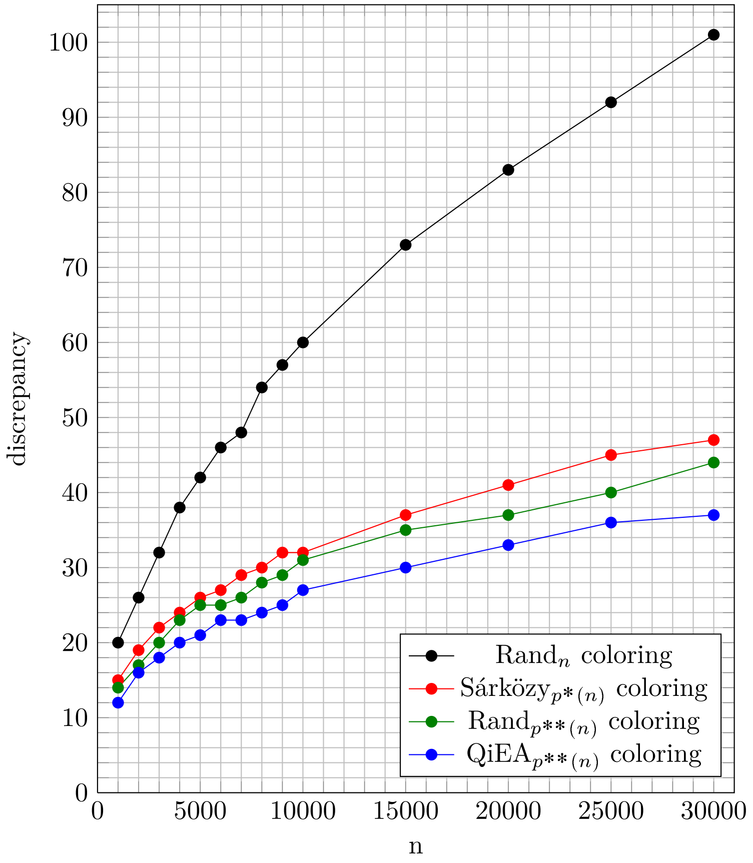


Figure 5.6. Discrepancy results for different algorithms.

Table 5.3. Two-coloring results of the modified Sárközy algorithm (using prime modules $p^*(n)$) and the single-group QiEA (using prime modules p^{**}), respectively.

n	Discrepancies	
	Sárközy	1-group QiEA
40000	54	42
50000	55	46
60000	59	46
70000	63	51
80000	66	52
90000	69	55
100000	71	55

Bibliography

[ABCC06] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem. A computational study.* Princeton Series in Applied Mathematics. Princeton, NJ: Princeton University Press. xi, 593 p., 2006.

[AFG00] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36(2):69–79, 2000. doi:10.1002/1097-0037(200009)36:2&lt;69::AID-NET1&gt;3.0.CO;2-Q.

[AFG01] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming A*, 90(3):475–506, 2001. doi:10.1007/s101070100218.

[ALED08] S. K. Amous, T. c. Loukil, S. Elaoud, and C. Dhaenens. A new genetic algorithm applied to the traveling salesman problem. *International Journal of Pure and Applied Mathematics*, 48(2):151–166, 2008.

[Alf04] H. K. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127:145–175, 2004. doi:10.1023/B:ANOR.0000019088.98647.e2.

[ANCK08] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008. doi:10.1016/j.ejor.2006.06.060.

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the Association for Computing Machinery*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

[AS08] N. Alon and J. H. Spencer. *The probabilistic method. With an appendix on the life and work of Paul Erdős. 3rd ed.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Hoboken, NJ: John Wiley & Sons. xv, 352 p., 2008.

[BB09] M. Bhattacharyya and A. K. Bandyopadhyay. Comparative study of some solution methods for traveling salesman problem using genetic algorithms. *Cybernetics and Systems*, 40(1):1–24, 2009. doi:10.1080/01969720802492967.

[BBBB72] R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk. *Statistical inference under order restrictions. The theory and application of isotonic regression.* Wiley Series in Probability and Mathematical Statistics. London etc.: John Wiley & Sons. XII,388 p., 1972.

[BC87] J. Beck and W. W. L. Chen. *Irregularities of distribution.* Cambridge Tracts in Mathematics, 89. Cambridge etc.: Cambridge University Press. XIV, 294 p., 1987.

[BDP09] G. S. S. Babu, D. B. Das, and C. Patvardhan. Solution of real-parameter optimization problems using novel quantum evolutionary algorithm with applications in power dispatch. In *IEEE Congress on Evolutionary Computation*, pages 1920–1927, 2009.

[Bec81] J. Beck. Roth's estimate of the discrepancy of integer sequences is nearly sharp. *Combinatorica*, 1:319–325, 1981. doi:10.1007/BF02579452.

[BHKK07] H.-J. Böckenhauer, J. Hromkovic, J. Kneis, and J. Kupke. The parameterized approximability of TSP with deadlines. *Theory of Computing Systems*, 41(3):431–444, 2007. doi:10.1007/s00224-007-1347-x.

[BS95] J. Beck and V. T. Sós. Discrepancy theory. Graham, R. L. (ed.) et al., Handbook of combinatorics. Vol. 1-2. Amsterdam: Elsevier (North-Holland). 1405-1446., 1995.

[BS11] N. Bansal and J. Spencer. Deterministic discrepancy minimization. Demetrescu, C. (ed.) et al., Algorithms – ESA 2011. 19th annual European symposium, Saarbrücken, Germany, September 5–9, 2011. Proceedings. Berlin: Springer. Lecture Notes in Computer Science 6942, 408-420., 2011. doi:10.1007/978-3-642-23719-5_35.

[BSW07] P. Brucker, Y. N. Sotskov, and F. Werner. Complexity of shop-scheduling problems with fixed number of jobs: a survey. *Mathematical Methods of Operations Research*, 65(3):461–481, 2007. doi:10.1007/s00186-006-0127-8.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press. xiii, 716 p., 2004.

[Car] Homepage of David L. Carroll GA: http://cuaerospace.com/carroll/ga.html.

[CDL04] T. C. E. Cheng, Q. Ding, and B. M. T. Lin. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152(1):1–13, 2004. doi:10.1016/S0377-2217(02)00909-8.

[Cha00] B. Chazelle. *The discrepancy method. Randomness and complexity.* Cambridge: Cambridge University Press. xviii, 463 p., 2000.

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In: Traub, J. F. (ed.). Symposium on New Directions and Recent Results in Algorithms and Complexity. Academic Press, Orlando, Fla., 1976.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing, Shaker Heights, Ohio 1971, 151-158., 1971.

[CP80] H. Crowder and M. W. Padberg. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26:495–509, 1980. doi:10.1287/mnsc.26.5.495.

[cpl] Homepage of IP solver Cplex: http://www.ilog.com/products/optimization/archive.cfm.

[Dak65] R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8:250–255, 1965. doi:10.1093/comjnl/8.3.250.

[Dan63] G. B. Dantzig. *Linear programming and extensions.* A RAND Corporation Research Study. Princeton, N.J.: Princeton University Press. XVI, 625 p. , 1963.

[Dav91] L. Davis, editor. *Handbook of genetic algorithms.* New York: Van Nostrand Reinhold. 385 p., 1991.

[Deb01] K. Deb. *Multi-objective optimization using evolutionary algorithms.* Chichester: Wiley. xix, 497 p., 2001.

[DFJ54] G. B. Dantzig, D. R. Fulkerson, and S. Johnson. Solution of a Large Scale Traveling Salesman Problem. *Operations Research*, 2:393–410, 1954.

[DPSK09] M. Defoin-Platel, S. Schliebs, and N. Kasabov. Quantum-inspired evolutionary algorithm: A multimodel eda. *IEEE Transactions on Evolutionary Computation*, 13(6):1218–1232, 2009.

[DS03] B. Doerr and A. Srivastav. Multicolour discrepancies. *Combinatorics, Probability & Computing*, 12(4):365–399, 2003.

[dSU10] R. F. da Silva and S. Urrutia. A general vns heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203–211, 2010.

[EO07] M. El Ouali. Das Travelling-Salesman-Problem mit mehreren Zeitfenstern. Diploma Thesis, Christian-Albrechts-Universität, Kiel, Germany, 2007.

[EP85] G. T. Evans and J. S. Parslow. A model of annual plankton cycles. *Biological Oceanography*, 3:328–347, 1985.

[ES74] P. Erdős and J. Spencer. *Probabilistic methods in combinatorics.* Budapest: Akademiai Kiado. 106 p., 1974.

[ES92] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In *FOGA*, pages 187–202, 1992.

[FH03] L. Fortnow and S. Homer. A short history of computational complexity. *Bulletin of the European Association for Theoretical Computer Science, EATCS*, 80:95–133, 2003.

[FL05] C. A. Floudas and X. Lin. Mixed integer linear programming in process scheduling: modeling, algorithms, and applications. *Annuals of Operations Research*, 139:131–162, 2005. doi:10.1007/s10479-005-3446-x.

[FLS] Homepage of Tour Planning Company *FLS (Fuhrpark Logistik Systeme GmbH)*: http://www.fls-service.de/Eng/.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution.* New York-London-Sydney: John Wiley and Sons, Inc. XII, 170 p. , 1966.

[GDC92] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6(4):333–362, 1992.

[GHLS98] M. Gendreau, A. Hertz, G. Laporte, and M. Stan. A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, 46(3):330–335, 1998.

[GJ78] M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of the Association for Computing Machinery*, 25:499–508, 1978. doi:10.1145/322077.322090.

[GMW81] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization.* London etc.: Academic Press, a Subsidiary of Harcourt Brace Jovanovich, Publishers. XVI, 401 p., 1981.

[Gom58] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958. doi:10.1090/S0002-9904-1958-10224-4.

[GP85] M. Grötschel and M. W. Padberg. Polyhedral theory. The traveling salesman problem, a guided tour of combinatorial optimization, 251-305., 1985.

[GP02] G. Gutin and A. P. Punnen, editors. *The traveling salesman problem and its variations.* Combinatorial Optimization. 12. Dordrecht: Kluwer Academic Publishers. xviii, 830 p., 2002.

[GSK98] C. P. Gomes, B. Selman, and H. A. Kautz. Boosting combinatorial search through randomization. In *AAAI/IAAI*, pages 431–437, 1998.

[Har02] S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49(5):433–448, 2002. doi:10.1002/nav.10029.

[HK02] K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6):580–593, 2002.

[HK03] K.-H. Han and J.-H. Kim. On setting the parameters of quantum-inspired evolutionary algorithm for practical application. In *IEEE Congress on Evolutionary Computation (1)*, pages 178–194, 2003.

[HK04] K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithms with a new termination criterion, H_ϵ gate, and two-phase scheme. *IEEE Transactions on Evolutionary Computation*, 8(2):156–169, 2004.

[Hol75] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975. URL: http://books.google.de/books?id=vk5RAAAAMAAJ.

[Hon72] S. Hong. A linear programming approach for the traveling salesman problem. Ph.D. Thesis, The Johns Hopkins University., 1972.

[HP11] M. Hojati and A. S. Patil. An integer linear programming-based heuristic for scheduling heterogeneous, part-time service employees. *European Journal of Operational Research*, 209(1):37–50, 2011. doi:10.1016/j.ejor.2010.09.004.

[JLN$^+$10] M. Jünger, T. M. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors. *50 years of integer programming 1958–2008. From the early years to the state-of-the-art. Papers based on the presentations at the special session at the 12th combinatorial optimization workshop AUSSOIS 2008, Aussois, France January 7–11, 2008. With DVD.* Berlin: Springer. xx, 801 p., 2010. doi:10.1007/978-3-540-68279-0.

[JM08] K. Jansen and M. Margraf. *Approximative algorithms and non-approximability. (Approximative Algorithmen und Nichtapproximierbarkeit.).* Berlin: de Gruyter. xv, 501 p., 2008.

[Kar75] R. M. Karp. Reducibility among combinatorial problems. *Kiberneticheskij Sbornik, Novaya Seriya*, 12:16–38, 1975.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984. doi:10.1007/BF02579150.

[KC09] D. Kincaid and W. Cheney. *Numerical analysis. Mathematics of scientific computing. Reprint of the 3rd ed. 2002 published by Brooks/Cole.* Pure and Applied Undergraduate Texts 2. Providence, RI: American Mathematical Society (AMS). xiv, 788 p., 2009.

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics, Doklady*, 20:191–194, 1979.

[KLPS07] A. W. J. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. R. Spieksma. Interval scheduling: a survey. *Naval Research Logistics*, 54(5):530–543, 2007. doi:10.1002/nav.20231.

[Kni97] P. Knieper. Discrepancy of arithmetic progressions. Doctoral Dissertation, Humboldt University Berlin, Department of Computer Science, 1997.

[Kos93] P. Kosmol. *Methods for the numerical treatment of nonlinear equations and optimization problems. (Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben.) 2., überarb. Aufl.* Teubner Studienbücher: Mathematik. Stuttgart: Teubner,. ix, 230 p., 1993.

[KS08] L. Kliemann and A. Srivastav. Parallel algorithms via the probabilistic method. Rajasekaran, S. (ed.) and Reif, J. (ed.), Handbook of parallel computing. Models algorithms, and applications. Chapman & Hall/CRC Computer and Information Science Series, chapter 18, 61 pages. Boca Raton, FL: Chapman & Hall/CRC. not consec. pag., 2008. doi:10.1201/9781420011296.

[LD60] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960. doi:10.2307/1910129.

[Lev73] L. A. Levin. Universal problems of full search. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[LK73] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *European Journal of Operational Research*, 21:498–516, 1973. doi:10.1287/opre.21.2.498.

[LL00] D. J. Leith and W. E. Leithead. Survey of gain-scheduling analysis and design. *International Journal of Control*, 73(11):1001–1025, 2000. doi:10.1080/002071700411304.

[LRS10] W. Lehmann, T. Rabe, and V. Sauerland. Scheduling: Objectives & Constraints. Technical Report, MINT GmbH, Germany, 98 pages, 2010.

[Mat99] J. Matoušek. *Geometric discrepancy. An illustrated guide.* Algorithms and Combinatorics. 18. Berlin: Springer. xi, 288 p., 1999.

[Mat00] J. Matoušek. Derandomization in computational geometry. Sack, J.-R. (ed.) et al., Handbook of computational geometry. Amsterdam: North-Holland. 559-595., 2000.

[MHS10] M. Müller-Hannemann and S. Schirra, editors. *Algorithm Engineering: Bridging the Gap between Algorithm Theory and Practice [outcome of a Dagstuhl Seminar]*, volume 5971 of *Lecture Notes in Computer Science*. Springer, 2010.

[MMP05] Y. Marinakis, A. Migdalas, and P. M. Pardalos. A hybrid genetic-GRASP algorithm using Lagrangean relaxation for the traveling salesman problem. *Journal of Combinatorial Optimization*, 10(4):311–326, 2005. doi:10.1007/s10878-005-4921-7.

[MNN94] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994. doi:10.1016/S0022-0000(05)80069-8.

[mpi] Homepage of MPICH2 open source MPI implementation: http://www.mcs.anl.gov/research/projects/mpich2/.

[MS96] J. Matoušek and J. Spencer. Discrepancy in arithmetic progressions. *Journal of the American Mathematical Society*, 9(1):195–204, 1996. doi:10.1090/S0894-0347-96-00175-0.

[OCC] Homepage of the OCCAM global ocean model: http://www.noc.soton.ac.uk/JRD/OCCAM/.

[OG99] A. Oschlies and V. Garcon. An eddy-permitting coupled physical-biological model of the North Atlantic. 1. Sensitivity to advection numerics and mixed layer physics. *Global Biogeochemical Cycles*, 13:135–160, 1999.

[OR00] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables.* Classics in Applied Mathematics. 30. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. xxvi, 572 p., 2000. doi:10.1137/1.9780898719468.

[PGPR99] G. Pesant, M. Gendreau, J.-Y. Potvin, and J.-M. Rousseau. On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem. *European Journal of Operational Research*, 117(2):253–263, 1999.

[PKS11] M. Pries, S. Koziel, and T. Slawig. Surrogate-Based Optimization of Climate Model Parameters Using Response Correction. *Journal of Computational Science*, 2:335–344, 2011.

[Pot96] J-Y. Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:339–370, 1996. doi:10.1007/BF02125403.

[PS98] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity. Corr. repr. of the 1982 original.* Mineola, NY: Dover Publications, Inc. xvi, 496 p., 1998.

[PS12] M. Pries and T. Slawig. Aggressive Space Mapping for the Optimization of a Marine Ecosystem Model. *International Journal of Mathematical Modeling and Numerical Optimization*, 3:98–116, 2012.

[Rab12] T. Rabe. Entwurf effizienter Algorithmen für die Lehrgangsplanung bei Fluggesellschaften. Diploma Thesis, Christian-Albrechts-Universität, Kiel, Germany, to appear, 2012.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Reihe Problemata. Frommann-Holzboog, 1973. URL: http://books.google.de/books?id=-WAQAQAAMAAJ.

[Ree03] C. Reeves. Genetic algorithms. Glover, F. (ed.) et al., Handbook of metaheuristics. Boston, MA: Kluwer Academic Publishers. International Series in Operations Research & Management Science. 57, 55-82., 2003. doi:10.1007/0-306-48056-5_3.

[Rot64] K. F. Roth. Remark concerning integer sequences. *Acta Arithmetica*, 9:257–260, 1964.

[RSL77] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6:563–581, 1977. doi:10.1137/0206041.

[RSS$^+$10] J. Rückelt, V. Sauerland, T. Slawig, A. Srivastav, C. Patvardhan, and B. Ward. Parameter optimization and uncertainty analysis in a model of oceanic CO_2 uptake using a hybrid algorithm and algorithmic differentiation. *Nonlinear Analysis B Real World Applications*, 11(5):3992–4009, 2010.

[RT87] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987. doi:10.1007/BF02579324.

[Sah74] S. Sahni. Computational related problems. *SIAM Journal on Computation*, 3:262–279, 1974.

[Sav85] M. W. P. Savelsberg. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.

[Sch81] H.-P. Schwefel. *Numerical optimization of computer models. Transl. from the German.* Chichester etc.: John Wiley & Sons. VII, 389 p., 1981.

[SMOK06] S. Suresh, V. Mani, S. N. Omkar, and H. J. Kim. Divisible load scheduling in distributed system with buffer constraints: genetic algorithm and linear programming approach. *International Journal of Parallel Emergent and Distributed Systems*, 21(5):303–321, 2006. doi:10.1080/17445760600567842.

[SO03] M. Schartau and A. Oschlies. Simultaneous data-based optimization of a 1D-ecosystem model at three locations in the North Atlantic: Part I - Method and parameter estimates. *Journal of Marine Research*, 61(6):765–793, 2003.

[Spe85] J. Spencer. Six standard deviations suffice. *Transactions of the American Mathematical Society*, 289:679–706, 1985. doi:10.2307/2000258.

[Sri01a] A. Srinivasan. New approaches to covering and packing problems. Kosaraju, Deborah, Proceedings of the 12th annual ACM-SIAM symposium on discrete algorithms. Washington, DC, USA, January 7-9, 2001. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. 567-576., 2001.

[Sri01b] A. Srivastav. Derandomization in combinatorial optimization. Rajasekaran, S. (ed.) et al., Handbook of randomized computing. Vols. 1, 2. Dordrecht: Kluwer Academic Publishers; 0-7923-6957-2 (v. 1); 0-7923-6958-0 (v. 2)). Comb. Optim. 9, 731-842., 2001.

[SSW00] N. V. Shakhlevich, Y. N. Sotskov, and F. Werner. Complexity of mixed shop scheduling problems: A survey. *European Journal of Operational Research*, 120(2):343–351, 2000. doi:10.1016/S0377-2217(99)00161-7.

[TB01] V. T'kindt and J.-C. Billaut. Multicriteria scheduling problems: a survey. *RAIRO, Operations Research*, 35(2):143–163, 2001. doi:10.1051/ro:2001109.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society. Second Series*, 42:230–265, 1936. doi:10.1112/plms/s2-42.1.230.

[VL90] J. Van Leeuwen, editor. *Algorithms and complexity. Handbook of theoretical computer science. Vol. A.* Amsterdam etc.: Elsevier Science Publishers; Cambridge, MA: The MIT Press. IX, 996 p., 1990.

[Weg05] I. Wegener. *Complexity theory. Exploring the limits of efficient algorithms. Translated from the German by Randall Pruim.* Berlin: Springer. xi, 308 p., 2005.

[YG10] X. Yu and M. Gen. *Introduction to evolutionary algorithms.* Decision Engineering. London: Springer. xvi, 418 p., 2010. doi:10.1007/978-1-84996-129-5.

[Zha11] G. Zhang. Quantum-inspired evolutionary algorithms: a survey and empirical study. *Journal of Heuristics*, 17(3):303–351, 2011. doi:10.1007/s10732-010-9136-0.