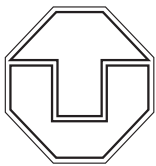


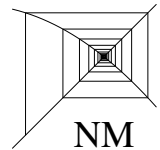
Ideas and Know-how in Modeling and Solving
Discrete Optimization Problems

18th Workshop on Discrete Optimization



Königstein, May 18–21, 2008

Institute of Numerical Mathematics
Technische Universität Dresden



Organizers

Andreas Fischer (Dresden)
Guntram Scheithauer (Dresden)
Gleb Belov (Dresden)

Dear Participant,

welcome to the Workshop on Discrete Optimization. We wish you a fruitful and pleasant stay in Königstein and very much hope you can take home some new ideas.

This paper contains remarks of general interest, the time schedule, the abstracts of the talks, and a list of the participants.

We would like to express our thanks to the Technische Universität Dresden, to the Institute of Numerical Mathematics and to the people who have helped us to prepare the workshop, in particular Mrs. Elke Krug. Please do not hesitate to contact us in the case of questions or problems.

Enjoy the meeting!

Andreas Fischer
Guntram Scheithauer
Gleb Belov

Contents

Conference Location	3
Schedule	4
Abstracts	6
List of Participants	33

Conference Location

Hotel "Lindenhof"
Gohrischer Straße 2
D-01824 Königstein / Sachsen
Tel.: +4935021 68243

Conference Program

Sunday, May 18, 2008

18.00	OPENING	Informal Get-Together in Königstein
-------	---------	-------------------------------------

Monday, May 19, 2008

10.00 – 11.00	I. Althöfer	Valleys of bad performance in discrete optimization problems
11.00 – 11.30	M. Dörnfelder	On the number of different alternative solution methods generated by the penalty method
11.30 – 12.00	R. Walter	Competitive Scheduling Scenarios
12.00	LUNCH	
14.00 – 14:30	H. Bräsel	LiSA version 3.0 and its application to open-shop problems with due dates
14.30 – 15.00	M. Andresen	Sequence implication classes in open-shop scheduling
15.00 – 15.30	Yu.N. Sotskov, N.M. Matsveichuk, N.G. Egorova	Schedule execution to minimize makespan for two-machine flowshop with uncertain processing times
15.30 – 16.00	COFFEE BREAK	
16.00 – 16.30	T. Starostina	Solving dynamic flow problem using simulation
16.30 – 17.00	C. Hemig, J. Zimmermann	Production planning in the final assembly of an automotive plant
17.00 – 17.30	L. Hempel	A review on history of computers

Tuesday, May 20, 2008

9.30 – 10.00	T. Romanova, Y. Stoyan, A. Krivylya, M. Zlotnik, G. Scheithauer	Covering of a polygonal region by rectangles
10.00 – 10.30	P.K. Shukla	An efficient, adaptive scalarization scheme for multi-objective discrete optimization
10.30 – 11.00	COFFEE BREAK	
11.00 – 11.30	D. Fanghänel, F. Liers	A fast exact algorithm for the optimum cooperation problem

11.30 – 12.00	J. Rietz	Special difficulties in the three-dimensional container loading problem
12.00	LUNCH	
14.00 – 17.00	DISCUSSIONS	

Wednesday, May 21, 2008

10.00 – 10.30	G. Scheithauer	Cutting and packing problems: presentation of a new book
10.30 – 11.00	G. Belov	An overview of lower bounds for orthogonal packing
11.00 – 11.30	H. Rohling, G. Belov	LP-based branching in the interval graph algorithm for orthogonal packing
11.30 – 12.00	M. Mesyagutov, G. Scheithauer, G. Belov	LP-based exact approach for 1D contiguous stock cutting
12.00	LUNCH	
14.00 – 15.00	CLOSING SESSION	

18th Workshop on Discrete Optimization

Königstein, May 18–21, 2008

Abstracts

Ingo Althöfer

Faculty of Mathematics and Computer Science
Friedrich-Schiller University Jena

Valleys of Bad Performance in Discrete Optimization Problems

Three rather different case studies are discussed to demonstrate the phenomenon of “**Valleys of Bad Performance**”

* C. Rose [4] analysed a majority approach for combinatorial optimization problems on n -dimensional Hamming cubes: Three good solutions are generated by local search. From them, a consensus solution is built, taking the majority componentwise. Starting from this consensus solution, another run of local search is performed. For large values of n the approach was (very) successful. However, **for intermediate dimensions n the results were worse** than those from an independent fourth run of local search.

* S. Kolassa [2], partly supported by S. Schwarz [3], analysed a two-step shortlisting procedure, which is performed by two independent non-perfect agents. Agent 1 reduces the original set of n candidates to a set of intermediate size k , and Agent 2 makes the final choice from this reduced list. It turns out that often there is an optimal intermediate size k^* , and between $k = 1$ and $k = k^*$ there is a **Valley of Bad Performance**.

* A. Irmer [1] is applying Monte Carlo methods in game tree search. When using nontrivial heuristics instead of completely random sampling, it can happen that **search with intermediate Monte Carlo parameters gives worse results** than search with Monte Carlo parameter = 0.

References

- [1] A. Irmer. On Phenomena in Monte-Carlo Game Tree Search. Diploma Thesis (in preparation), FSU Jena, Faculty of Mathematics and Computer Science, 2008.
- [2] S. Kolassa. Multi-Step Shortlisting by Imperfect Experts. Doctoral dissertation, FSU Jena, Faculty of Mathematics and Computer Science, 2004. Published in 3- Hirn-Verlag, Lage (Germany), <http://www.3-hirn-verlag.de/books.html>
- [3] S. Kolassa and S. Schwarz. Two-step drawing from urns. In “Operations Research Proceedings 2004” (Eds. H. Fleuren, D. den Hertog and P. Kort), Springer, Berlin and Heidelberg, pp. 313-318. ISBN 978-3-540-24274-1 (Print), 978-3-540-27679-1 (Online).
- [4] C. Rose. Mehrheitsbildung in der kombinatorischen Optimierung. Doctoral dissertation (in German), FSU Jena, Faculty of Mathematics and Computer Science, 2000.

Michael Andresen

Otto-von-Guericke-Universität Magdeburg

Sequence implication classes in open-shop scheduling

Solutions of scheduling problems are called *sequences*. They can be described using comparability graphs. A *comparability graph* is a graph whose edges can be oriented transitively. Every *transitive orientation* of a graph corresponding to a scheduling problem corresponds to a solution to this problem. A common concept for describing comparability graphs is the consideration of a so-called Γ -*relation* on the edge set E . The transitive closure of this relation is an equivalence relation whose equivalence classes are called *implication classes*.

Sequences on the 'efficiency frontier' are of special interest. These sequences are called *irreducible*. For every given choice of processing times the set of all irreducible sequences contains an optimal solution.

For open-shop scheduling (no presettings for job order or machine order) with makespan criterion the notion of comparability graphs can be used to decide in polynomial time whether a sequence A reduces some other sequence B (see Bräsel et al. [1]). However, it is an open problem to decide in polynomial time whether a given sequence A is irreducible.

To tackle this problem Willenius [2] introduced the notion of *sequence implication classes* as a substructure of implication classes. These sequence implication classes will be introduced and their importance outlined.

References

- [1] Bräsel, Heidemarie; Harborth, Martin; Tautenhahn, Thomas und Willenius, Per (1999). On the set of solutions of an open shop Problem. *Ann. Oper. Res.* (1999) **92** 241-263.
- [2] Willenius, Per (2000). *Irreduzibilitätstheorie bei Shop-Scheduling-Problemen*. Dissertationsschrift. Shaker Verlag, 2000.

Gleb Belov

Institut für Numerische Mathematik
Technische Universität Dresden

An overview of lower bounds for orthogonal packing

Consider the d -dimensional orthogonal packing problem (OPP- d): given a set of orthogonal objects, n items and a container, it is to answer if the items can be packed in the container. No rotation is allowed.

We review various bounds from the literature and present some comparisons.

LiSA Version 3.0 and its Application to Open-shop Problems with Duedates

LiSA is a software package for solving deterministic scheduling problems in terms of the well-known $\alpha | \beta | \gamma$ - notation, where α describes the machine environment, β contains additionally constraints and γ gives the objective function. Because of the internal data-structure, LiSA is particularly suitable for shop problems. In the talk we give an overview on the modular structure of LiSA Version 3.0, especially on the news in comparison to former versions. Now we are able to use most of the algorithms outside of LiSA. This can be done by using a so-called auto-alg file. This file contains the problem description and the parameters to generate the input data and finally, step by step the calls of the algorithms which have to be applied to the problem. Moreover, it is possible, to start a new algorithm, for instance simulated annealing, from the best schedule calculated earlier by different constructive heuristics. Hybrid algorithms can also be composed, for instance by means of a list of different metaheuristics.

To demonstrate the new powerful tools, we investigate open-shop problems with release dates and the objective function $\sum w_i T_i$, i.e. the sum of the weighted tardiness has to be minimized. Each considered problem can be described by a tuple w, p, r, d for the weights w_i , the processing times p_{ij} , the release dates r_i and the due-dates d_i . We construct 24 different problems by fixing the parameters. For each problem we consider 16 different formats $n \times m$, $n, m \in \{10, 15, 20, 30\}$, where in all cases 20 instances are generated. For every instance we first apply some constructive heuristics and then we continue with different simulated annealing algorithms. Here we use our experience on the properties of the open-shop problem with minimizing mean flow time (see [1] [2]). Finally we compare the results with the quality of genetic algorithms. In total, we apply more than 100 algorithms to each instance. The data interpretation seems to be difficult. Nevertheless we will give an overview on the main results of our computational experiments, contained in Andresen et al. [3].

LiSA-Version 3.0 is available under <http://lisa.math.uni-magdeburg.de>, the handbook will be finished in summer 2008.

References

- [1] Bräsel, H.; Herms, A.; Mörig, M.; Tautenhahn, T.; Tusch, J.; Werner, F., *Heuristic Constructive Algorithms for Open Shop Scheduling to Minimize Mean Flow Time*, to appear in European Journal of Operational Research; published online.
- [2] Andresen, M.; Bräsel, H.; Mörig, M.; Tusch, J.; Werner, F.; Willenius, P., *Simulated Annealing and Genetic Algorithms for Minimizing Mean Flow Time in an Open Shop*, to appear in Mathematical and Computer Modelling, published online.
- [3] Andresen, M.; Bräsel, H.; Plauschin, M.; Werner, F., *Using Simulated Annealing for Open Shop Scheduling with Sum Criteria*, to appear as book chapter in Global Optimization: Focus on Simulated Annealing, ISBN 978-3-902613-33-2.

Martin Dörnfelder

Faculty of Mathematics and Computer Science
Friedrich-Schiller-University Jena

On the Number of Different Alternative Solutions Generated by the Penalty Method

In discrete optimization problems one is usually interested in generating an optimal solution. Nevertheless in many practical cases it is advisable to generate not only an optimal solution but also alternative solutions to be able to react appropriately to unforeseen events. Here a simple approach is to take the 2nd best solution as the alternative one. Typically it turns out that this solution is pretty similar to the best one, which is a disadvantage in practical sense. A better strategy for generating real alternatives is the penalty method. I applied penalty method to graph problems with an objective function of sum type. Sum type means, that the objective function is the sum of the weights of all edges in the solution. Here it works as follows:

- Create an optimal solution.
- Multiply all weights of edges which are in the best solution with factor $(1 + \varepsilon)$ for $\varepsilon > 0$.
- Create an optimal solution in the graph with the modified weights. This solution is called ε -alternative.

I studied the problem how many different alternative solutions can be generated for shortest path problems and for matroid problems. Without loss of generality we assume that the graph has n vertices and is complete. It is known that for shortest path problems examples exist with $\Omega(n^2)$ alternative solutions which are optimal for different penalty parameters. I could show, that there are only $O(n^2)$ alternative solutions for shortest path problems on directed acyclic graphs. For matroid problems like minimum spanning trees it could be shown, that there are only $O(n)$ alternative solutions which can be optimal for any penalty parameter.

A Fast Exact Algorithm for the Optimum Cooperation Problem

In this work, we will deal with the following optimization problem. Suppose the benefit of cooperation between two vertices, say, researchers, is represented by the weight of the edge joining them. Furthermore, there is a unit gain for each component, say, research project. We search for an optimal cooperation, i.e., want to decide which researchers should collaborate in order to maximize the total benefit (see Auriac et al. [1]).

In graph-theoretic terms, the problem can be stated as follows. Let a graph $G = (V, E)$ with edge weights $w_e \in \mathbf{R}$ for the edges $e \in E$ be given. We want to solve the problem

$$\max\{c_G(A) + w(A) : A \subseteq E\}, \quad (1)$$

where $w(A) = \sum_{e \in A} w_e$ and $c_G(A)$ is the number of connected components of the induced graph $G(A) = (V, A)$.

The problem was first mentioned in the literature by Cunningham [5] in the context of determining optimum attacks in networks. At this the edge weight w_e can be interpreted as a measure for the effort required by an attacker to destroy edge e . The task is to minimize the difference between the effort of destroying a set of edges and the number of newly generated components of the graph.

Another relevant application is the separation of partition inequalities, as introduced by Baiou, Barahona and Mahjoub [2]. Given a partition $\{S_1, \dots, S_p\}$ of the node set V , we denote by $\delta(S_1, \dots, S_p)$ the set of all edges having endnodes in different sets of the partition. Then, for given real numbers a and b , the inequality $w(\delta(S_1, \dots, S_p)) \leq ap + b$ is called partition inequality. Partition inequalities arise as valid inequalities for a number of combinatorial problems. In order to use these inequalities inside a cutting plane algorithm, we have to solve the separation problem that, given edge weights $w_e \geq 0$, returns a partition violating the inequality, if it exists. Baiou et al. [2] show that the separation problem can be solved by computing an optimal solution of (1).

An important model in statistical physics is the so-called Potts model (see Hartmann/Rieger [7]). It was introduced as a generalization of the so-called Ising model to describe several physical systems. It is a model on a graph where the vertices are assigned (spin) variables that each can take values between $\{1, \dots, q\}$. Interactions between pairs of spins may be present. The aim is to compute the so-called partition function that completely encodes the physics of the system. It depends on the number q and the interactions between the spins. For many relevant physics systems, computing the partition function is a difficult task. However, as pointed out by Juhasz, Rieger, Iglói [8] and Anglès d'Auriac, Iglói, Preissmann and Sebö [1], for big numbers q , determining the dominant contribution in the Potts partition function amounts to solving a problem of type (1).

Several solution algorithms have been presented in the literature. As it is not hard to see that the function to be maximized is supermodular, any algorithm for submodular function minimization could be used to solve the problem. By now several polynomial algorithms [4; 6] are known to solve this task. However, the specific properties of the problem allow the usage of algorithms with better worst-case asymptotic running time.

Cunningham [5] developed the first combinatorial algorithm with polynomial running time for the solution of the optimum attack problem that is based on $|E|$ minimum cut computations in

an associated network. Thereafter, the worst-case running time of the algorithm was decreased to only $|V| - 1$ minimum cut computations by Baiou, Barahona and Mahjoub, and Barahona [2; 3]. Anglès d'Auriac et al. [1] built upon the existing work and presented an algorithm that also needs $|V| - 1$ minimum cut computations but is easier to implement. They presented some experimental results for instances coming from the physics application.

In the talk we present an algorithm that is based on the one of Anglès d'Auriac et al. but has a better average running time, as by graph-theoretic considerations only a fraction of the minimum cut computations are necessary in practice. We also provide optimality conditions and theoretical results that prove the correctness of our algorithm. We discuss details of our implementation and present experimental results on instances coming from the physics applications. Furthermore, we compare our algorithm with the original method of Anglès d'Auriac et al. It turns out that the running times can be reduced considerably for many practical instances.

References

- [1] J.-Ch. Anglès d'Auriac, F. Iglói, M. Preissmann, and A. Sebö, *Optimal cooperation and submodularity for computing Potts' partition functions with a large number of states*, J. Phys. A: Math. Gen. 35 (2002) pp. 6973-6983
- [2] M. Baiou, F. Barahona, R. Mahjoub, *Separation of partition inequalities*, Math. of Operations Research, vol. 25, no. 2 (2000) pp. 243-254
- [3] F. Barahona, *Separating from the dominant of the spanning tree polytope*, Operations Research Letters, vol. 12 (1992) pp. 201-203
- [4] S. T. McCormick, *Submodular Function Minimization*, In: K. Aardal et al., Eds., *Discrete Optimization: Handbooks in Operations Research and Management Science, vol.12*, Elsevier (2005) pp. 321-391
- [5] W. H. Cunningham, *Optimal Attack and Reinforcement of a Network*, Journal of the Association for Computing Machinery, vol. 32, no. 3 (1985) pp. 549-561
- [6] S. Fujishige, *Submodular Functions and Optimization*, Annals of Discrete Mathematics Vol.58, Elsevier, Amsterdam, 2005
- [7] A. K. Hartmann, H. Rieger, *New Optimization Algorithms in Physics*, Wiley-VHC, Berlin, 2004
- [8] R. Juhasz, H. Rieger, F. Iglói, *The random-bond Potts model in the large- q limit*, Phys. Rev. E, Vol. 64, 056122 (2001)

Production Planning in the Final Assembly of an Automotive Plant

1 Introduction

The automotive market has been a sellers' market in the middle of the last century and has altered to a buyers' market in the 80s and 90s. The demand for automobiles has changed from the purpose of gaining access to personal mobility to an expression of the buyer's individuality.

The turnaround in the reception of automobiles carried out by the consumers has forced the automotive OEMs to broaden the number of variants. So e.g. BMW produces thousands of variants out of nine models with about 10^{32} variants theoretically available (cf. [11]). These possibilities can only be provided by installing a flexible manufacturing system, which is surveyed e.g. in [4] and [13]. We annotate that for the automotive industry two types of flexibility are mainly relevant, namely product flexibility (cf. e.g. [14]) and volume flexibility (cf. e.g. [9], [8], [1]). Installing an adequate product flexibility and adjustment screws for the volume flexibility with respect to the given demands' forecast is a task of the strategic production planning level (cf. e.g. [6]). Our approach is located on the tactical level and utilizes the given flexibility instruments.

The automotive industry has recognized that the tactical planning of its production capacities and staff size by hand and experience is not efficient due to the working time required for such a planning, the great variety of adjustment options, and the various influences on the production systems. Therefore, in cooperation with our industrial partner we develop a software tool to support the production and staff planners in several plants. The resulting possibility of an immediate application to problems of real-world size and with real-world data enables us to include requirements from the cutting edge of automotive production planning into our research.

After a detailed problem formulation in Chapter 2, we present our solution approach in Chapter 3. To handle the arising complexity for the case of parallel production lines we disaggregate our holistic approach and focus on one of the two subproblems to be solved and close by some conclusions and an outlook on further research.

2 Problem Formulation and Model Building

A typical automotive plant mainly consists of a body shop, a paint shop, and a final assembly. In each shop there may exist several parallel production lines that are organized as a continuous flow production system. On the lines miscellaneous types of products, called models, are manufactured. Each line can either be a solitary line (i.e. only one product can be produced) or a product flexible line (i.e. several different products can be produced). Buffers of limited capacities between the shops can store intermediate products to decouple the production in subsequent shops. In this paper we focus on a final assembly with L parallel production lines where some or all of M products can be produced by more than one line.

In particular we investigate the interrelation of the production time, the production speed and the needed amount of workforce on the particular production lines to fulfill a given demand for each product in each period of a planning horizon T . The planning horizon covers a whole life cycle of

a product, i.e. three to six years, where a period has a length of a week or a month. Furthermore, we examine the dependencies of the parallel lines regarding the interchange of staff as well as the distribution of workload. The objective of our approach is to find a cost optimal solution taking into account technical and labor restrictions as well as the given flexibility instruments of the underlying shop.

The following sections describe the decisions of our mathematical model that have to be made for each period. Simultaneously, we introduce the most important restrictions and finalize the chapter by presenting an adequate cost function.

2.1 Shift Model, Cycle Time, and Staff Demand

The fundamental decisions concerning the objective target in the final assembly are the shift model $sm_{lt} \in \mathcal{SM}_l$ and the cycle time $ct_{lt} \in \mathcal{CT}_l$ for each period t and each line l , where the sets \mathcal{SM}_l and \mathcal{CT}_l contain all eligible shift models and cycle times of line l , respectively. These two decisions restrict the feasible region of all other decisions significantly.

Next to several cost values, e.g. shift allowances, a selected shift model sm_{lt} provides a certain amount of production time in period t on line l whereas the cycle time ct_{lt} determines the number of stations that have to be installed at the line. The number of stations is equal to the numbers of workers employed directly at the line which are supplemented by some additional workers, e.g. foremen.

The selection of the shift model and the cycle time on each line is independent of each other and determines the production capacity of each line as well as the amount of required workers. In principle we can choose both of them arbitrarily every period, but a change of the shift model as well as the cycle time requires significant organizational and technical changes. Consequently, a change is allowed to occur only once in a given number of subsequent periods SM_l^{min} and CT_l^{min} , respectively. To minimize idle capacity it is possible to cancel single shifts little by little as long as the capacity exceeds the given demand for products. The installed capacity restricts the production of the given demand which is described in the following section.

2.2 Demand and Production

Each period t the final assembly has to produce a certain amount D_{mt} for each product m allowing a given relative deviation D^V . The decision how many units of product m are produced on line l in period t is denoted by variables $p_{m lt}$ with

$$D_{mt} \cdot (1 - D^V) \leq \sum_{l=1}^L p_{m lt} \leq D_{mt} \cdot (1 + D^V) \quad \forall m = 1, \dots, M \quad \forall t = 0, \dots, T - 1. \quad (2)$$

In other words, there is a minimal demand $D_{mt} \cdot (1 - D^V)$ that has to be met and an additional one that may be met up to the maximum demand $D_{mt} \cdot (1 + D^V)$.

Allowing some deviation from the given demand means to deduce an over- or underproduction Δp_{mt} where these variables contain the differences between production and demand accumulated over all periods before t .

If two (or more) products are produced on one line, we can establish a preferred mix ratio $mix_{l,m,ct}^{pref}$ of the products for each cycle time so that all stations of the line are nearly equally loaded (cf. e.g. [10]). Obviously, summing up $mix_{l,m,ct}^{pref}$ over all models m equals 1 for each line l and each cycle time ct . It is allowed to deviate from $mix_{l,m,ct}^{pref}$ in a tight range, and each product m must use the capacity of line l at least at a given minimal proportion $mix_{l,m,ct}^{min}$. Vice versa, we

can calculate the maximal proportion for all models m that can be produced on line l as

$$mix_{l,m,ct}^{max} = 1 - \sum_{\substack{n=1 \\ n \neq m}}^M mix_{l,n,ct}^{min} \quad \forall l = 1, \dots, L, \quad \forall m = 1, \dots, M, \quad \forall ct \in \mathcal{CT}_l. \quad (3)$$

If product m cannot be produced on line l using cycle time ct , we set $mix_{l,m,ct}^{min} = mix_{l,m,ct}^{pref} = mix_{l,m,ct}^{max} = 0$. Vice versa, if l is a solitary line for product m , the three values are set equal to 1.

2.3 Staff

The labor requirements at line l in period t arising from the choice of sm_{lt} and ct_{lt} can be met by permanent and temporary staff ps_{lt} and ts_{lt} . Typically, permanent employees can only be dismissed at a few points of time (e.g. at the end of a quarter) while contracts with temporary ones can expire at the end of any period. Permanent employees are skilled superior to their temporary colleagues and are evidently necessary for a permanently high output quality. To avoid frequent dismissals and hirings due to changes of the demand for workforce, permanent workers can be displaced between the production lines, and for each line a so called working hours summary is installed. On the working hours summary of a line the workers accumulate their over- and undertime of each period to provide flexible working hours (cf. e.g. [5]) where the values whs_{lt} —and therefore the installed flexibility—are bounded by in-company-agreements and labor legislation. Due to the tactical planning horizon, we do not examine every single worker and associated summary, but the installed workforce at each line and the average of all workers' summaries currently employed at that line.

2.4 Objective Function

Our objective is to minimize a cost function which is given as the sum of the staff, production, and changing costs over all periods. Due to the tactical planning horizon we discount the costs with an interest rate of α . The staff costs C_t^S are composed of the basic wages for the temporary as well as for the permanent employees, shift premiums, social surcharges, and costs for organizational modifications deriving from their hiring, dismissal and displacement to other lines. The production costs C_t^P are the sum of the variable production cost (see Section 2.2) over all lines and products. The changing costs C_t^C are invoked by changing the shift model or the cycle time on a line and are modeled as an affine-linear function in the number of workers currently employed at the considered line. So the objective function that is to minimize can be written as

$$\sum_{t=0}^{T-1} (C_t^S + C_t^P + C_t^C) \cdot e^{-\alpha t}. \quad (4)$$

Obviously, the cost function (4) is separable and the restrictions concerning the working hours summary are nonlinear. For these reasons we selected a Dynamic Programming approach (cf. e.g. [2], [3]) to solve the outlined problem.

3 A Solution Approach using Dynamic Programming

3.1 Basic Approach

The main elements of any Dynamic Programming approach are states, decisions, a separable objective function, and transformation functions. In the following, we introduce these elements for the described planning problem.

A state z_t associated with some period t is characterized by the accumulated over- or underproduction Δp_{mt} as well as a number of line-specific state variables. These variables are the actually chosen shift models sm_{lt} and cycle times ct_{lt} , the number of periods elapsed since their last changes s_{lt}^{sm} and s_{lt}^{ct} , respectively, the numbers of permanent and temporary employees ps_{lt} and ts_{lt} , respectively, as well as their working hours summaries whs_{lt} .

A decision $x_t(z_t)$ is an element of the set of all possible decisions in state z_t , $X_t(z_t)$, and contains the determination of the shift models \widehat{sm}_{lt} and the cycle times \widehat{ct}_{lt} , the numbers of hirings and dismissals of permanent and temporary employees, Δps_{lt} and Δts_{lt} , the numbers of displacements of permanent workers, ds_{lkt} , as well as the decisions about the number of products manufactured on a certain line, p_{mkt} .

Given some state z_t and an associated decision $x_t(z_t)$, the transformation function $g_t(x_t(z_t), z_t)$ generates a new state z_{t+1} of the next period $t + 1$. The chosen shift models and cycle times are taken from $x_t(z_t)$ by $sm_{l,t+1} = \widehat{sm}_{lt}$ and $ct_{l,t+1} = \widehat{ct}_{lt}$, respectively. The number of employees and the accumulated over- or underproduction are modified using the following balance equations

$$ps_{l,t+1} = ps_{lt} + \Delta ps_{lt} + \sum_{k=1}^L (ds_{klt} - ds_{lkt}) \forall l = 1, \dots, L, \quad \forall t = 0, \dots, T - 1, \quad (5)$$

$$ts_{l,t+1} = ts_{lt} + \Delta ts_{lt} \forall l = 1, \dots, L, \quad \forall t = 0, \dots, T - 1, \text{ and} \quad (6)$$

$$\Delta p_{m,t+1} = \Delta p_{mt} + \sum_{l=1}^L p_{mkt} - D_{mt} \forall m = 1, \dots, M, \quad \forall t = 0, \dots, T - 1. \quad (7)$$

The value for $s_{l,t+1}^{sm}$ —and in analogy for $s_{l,t+1}^{ct}$ —is set to $s_{lt}^{sm} + 1$ if the shift model on line l does not change from period t to $t + 1$ and set to 0 otherwise. The possible values for s_{lt}^{sm} and s_{lt}^{ct} for all $t = 0, \dots, T$ have an upper bound of SM_l^{min} and CT_l^{min} .

Cost function (4), introduced in Section 2.4, is separable in the periods and, therefore, the Bellman equation for a forward recursion can be written for all $t = 0, \dots, T - 1$ as

$$F_{t+1}^*(z_{t+1}) = \min_{\substack{x_t(z_t) \in X_t(z_t) \\ z_{t+1} = g_t(x_t(z_t), z_t)}} ((C_t^S + C_t^P + C_t^C) \cdot e^{-\alpha t} + F_t^*(z_t)) \quad (8)$$

Due to the tactical planning horizon and the huge sizes of the feasible decision and state space it is reasonable and necessary to merge similar states to one. To give an example, we consider only states that—*ceteris paribus*—differ in the number of hired permanent workers in discrete steps of 20. Using this idea we can assure the operative usability of the proposed approach for the case of one single production line. In the case of parallel lines this simplification does not suffice to control the increasing complexity. For this reason, we adopt an idea by Schneeweiß (cf. e.g. [12]) and disaggregate the approach into two levels, called Top and Bottom Level.

Both the distribution of the workload between the lines as well as the decisions concerning the staff are dependent on the chosen shift models and cycle times. Furthermore, the latter one is dependent on the former, but not the other way round. In other words, we can disaggregate a

decision's construction: First, we determine the shift models and cycle times on all lines. Second, we distribute the workload between the lines, and third, we specify the hirings, dismissals, as well as displacements of staff. In the following, we concentrate on the distribution of workload.

3.2 Distribution of Workload

The distribution of workload in some period t is restricted by two essentials: First, the choice of the shift model and the cycle time on every line determines the total capacity as well as the minimal and maximal capacity for each product (see Sections 2.1 and 2.2). Second, the minimal and additional demand for each product is given as input parameters.

Given some feasible combination of shift models and cycle times for each line we assign the minimal fractions of the capacity to the demand of the related products. This assignment is always possible otherwise the corresponding combination of shift models and cycle times would be infeasible. The remaining workload to be distributed is significantly smaller than before the assignment and will be allocated to the remaining capacities by modeling and solving the problem as a classical transportation problem (cf. e.g. [7]) where each line asks for and each product supplies some portion of the workload. The transportation unit costs are derived from the variable production costs per unit $c_{m,l,ct}^p$.

In particular, from each product, e.g. denoted with A , we generate two suppliers of our transportation problem, namely A_{min} offering the remaining minimal demand to be distributed after the considerations made above, and A_{add} offering the remaining additional demand of A .

Similarly to the generation of the suppliers, we separate every production line into as many destinations as products can be produced on that line. We restrict ourselves to just two products per line, but the approach can easily be extended to more products. In the following we model the preferred mix ratio of the selected cycle time on this line and its allowed deviation as mentioned in Section 2.2. Figure 1 illustrates the capacity of a line, denoted by 1, on which the products A and B can be produced. The dotted line illustrates the preferred mix ratio, the left and right hatched rectangles represent the minimal capacity for A and B that is filled with the accordant product's demand.

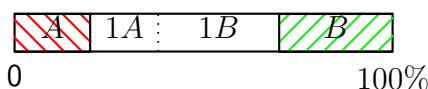


Figure 1: Distribution of the capacity between products A and B

Since the additional demand need not to be produced entirely, we introduce a dummy line to balance the transportation problem and to model unproduced additional demand.

A solution of the proposed transportation problem provides the information whether the remaining, unmarked areas—the yet unused capacity—should be filled with production workload of product A or B . Those areas' labels (see Figure 1) indicate the preferred model to be produced there to achieve the preferred mix ratio. The capacity marked with $1A$ and $1B$ should be filled with demand of product A and B , respectively, so that the preferred mix ratio is met. To achieve this preferred mix ratio we set the transportation costs per unit adequately, i.e. the higher the cost the less is the preference to transport the correspondent supply to the respective destination. If a line cannot produce a product the transportation unit cost is set to a sufficiently large value $BigM$.

Specifying the transportation unit costs, we do not differentiate between the two suppliers A_{min} and A_{add} , except for the dummy line. The unit cost for transporting from A_{min} to the dummy line is also set to $BigM$ because the minimal demand must always be produced on some real lines. We price the transportation from A_{add} to the dummy line with a cost value

greater than all transportation unit cost to any real line, but smaller than $BigM$. We establish those transportation costs per unit to introduce a preference order between all products regarding the production of the additional demand, for instance, dependent on the accumulated over- or underproduction.

We solve the transportation problem, add the workload already distributed, and obtain the values for p_{mlt} for all products m and all lines l . The decision still pending is the one concerning the staff. This is described in the following section.

3.3 Staff Planning

In order to reduce the computational effort we do not enumerate all feasible decisions concerning the staff. For each partial decision generated as described in Section 3.2 we build up exactly one decision for the staff. This restrictive shortening of considered decisions has to be compensated to avoid infeasibility in further periods. Hence, we incorporate increasing staff demand in the following periods as well as the necessity to take action for a feasible working hours summary.

The determination of the decisions concerning the staff is the last step to complete a decision of the Top Level. We can execute the algorithm of the Top Level as described and obtain an "optimal" solution keeping in mind that we used a heuristic for the decisions concerning the staff.

To improve the solution we start over the Dynamic Programming, fix the determined decision for the shift model, cycle time, and production program and enumerate all meaningful decisions concerning the staff as described in Section 3.1.

Altogether, we complete the decisions adopted from the Top Level by those generated on the Bottom Level and determine an optimal policy keeping in mind the disaggregation.

4 Conclusions and Outlook

In this paper we presented a Dynamic Programming approach to solve the problem of simultaneous tactical production and staff planning in the final assembly of an automotive plant. We focused on achieving tractability even for the case of parallel production lines that are not independent of each other and disaggregated the solution approach into two levels. On the Top Level we modeled the problem of distributing the production workload between the production lines as a transportation problem where the planning of the staff size is solved approximately. On the Bottom Level we fixed all decisions beside the staff planning and calculated an optimal staff planning solution.

Areas of future research and development are: First, to integrate the techniques described in this paper into our industrial partner's software tool which is already able to manage subsequent shops each with one production line and intermediate buffers in between them and second, to improve the Bottom level of the approach. More precisely, the Bottom Level improves the Top Level's solution just slightly and, therefore, the improvement of the Bottom Level should be done with respect to reduction of time consumption. For instance, a Local Search heuristic, starting with the solution given from the Top Level, might result in such a significant improvement.

References

- [1] G. Askar, T. Sillekens, L. Suhl, and J. Zimmermann. Flexibility planning in automotive plants. In H.-O. Günther, D.C. Mattfeld, and L. Suhl, editors, *Management logistischer Netzwerke*, pages 235–255. Physica, Heidelberg, 2007.

- [2] R.E. Bellman. *Dynamic Programming*. Princeton, University Press, New Jersey, 1957.
- [3] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, 2000.
- [4] J. Browne, D. Dubois, K. Rathmill, S.P. Sethi, and K.E. Stecke. Classification of flexible manufacturing systems. *The FMS Magazine*, 2:114–117, 1984.
- [5] L. Delsen, D. Bosworth, H. Groß, and R. Muñoz de Bustillo y Llorente, editors. *Operating Hours and Working Time*. Physica–Verlag, Heidelberg, New York, 2007.
- [6] P.C. Henrich. *Strategische Gestaltung von Produktionssystemen in der Automobilindustrie*. PhD thesis, University of Augsburg, 2002.
- [7] F.L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, 20:224–230, 1941.
- [8] E.P. Jack and A. Raturi. Sources of volume flexibility and their impact on performance. *Journal of Operations Management*, 20:519–548, 2002.
- [9] M. Khouja. An aggregate production planning framework for the evaluation of volume flexibility. *Production Planning and Control*, 9(2):127–137, 1998.
- [10] N. Leopold. *Ein Planungsverfahren zur Kapazitätsabstimmung für Modell–Mix–Montagelinien am Beispiel einer Automobil–Endmontage*. Number 130 in IPA–IAO – Forschung und Praxis. Springer–Verlag Berlin, 1997.
- [11] H. Meyr. Supply chain planning in the German automotive industry. *OR Spectrum*, 26(4): 447–470, 2004.
- [12] C. Schneeweiß. Hierarchical planning in organizations: Elements of a general theory. *International Journal of Production Economics*, 56–57:547–556, 1998.
- [13] A.K. Sethi and S.P. Sethi. Flexibility in manufacturing: A survey. *International Journal of Flexible Manufacturing Systems*, 2(4):289–328, 1990.
- [14] N. Slack. The flexibility of manufacturing systems. *International Journal of Operations & Production Management*, 7(4):35–45, 1987.

LP-based exact approach for the 1D contiguous cutting-stock problem

Given a set of indexes $I = \{1, 2, \dots, m\}$ of m rectangular items, each having width w_i and height h_i ($i \in I$), and a strip of width W and infinite height, the two-dimensional strip-packing problem (2SP) consists of orthogonally packing all the items, without overlapping, into the strip by minimizing the overall height of the packing. We assume that the items have fixed orientation, i.e., they cannot be rotated.

Let us consider a problem which is closely related to the latter. Given an instance of 2SP we formulate the following relaxed problem. Consider, for each $i \in I$, h_i items of size $w_i \times 1$. For each $i \in I$, all items $w_i \times 1$ have to be packed contiguously in the H -direction in order to find a packing which minimizes the height. Each item can be packed only once into one stock. This problem is known as the one-dimensional contiguous stock-cutting problem (1CSC).

Thus, each feasible solution of 2SP is a feasible solution of the corresponding 1CSC problem, but not vice versa. An optimum of 2SP is not necessarily an optimum of the corresponding 1CSC. According to the definition of 1CSC, the height of an optimum (or any lower bound for the latter) of 1CSC is also a lower bound of the corresponding 2SP. The domination of this bound over material bound is obvious. The 1CSC problem has also many applications, e.g., in scheduling.

We propose an exact approach to solve 1CSC. The approach is based on a branch & bound method on item sequence permutations with different search strategies. It is quite similar to the one from [2] where 1CSC (called 1CBP, 1D contiguous bin packing) was used as a lower bound for 2SP. The branching is organized in such a way that each node is divided into subproblems according to the following principle. Each child differs from the parent that in the child node one of the item types $i \in I$ is fixed in some free position. Thereby, the number of children of each node is equal to the number of free items, i.e., the root node contains m children, nodes of depth one contain $m-1$ children, etc.

For the calculation of a lower bound we use the continuous relaxation of the one-dimensional cutting-stock problem with multiple stock lengths (1CSPM).

Let a partial solution be given where rectangles \bar{I} are already packed, see Figure 2. The unused area with the width of λ_k , and the height of u_k corresponds to a set of size u_k of stock materials with the width λ_k and unit height. Thus, at the disposal of the remaining rectangles $\bar{I} = I \setminus \tilde{I}$ for cutting, we have q types of stock materials with widths λ_k where the largest stock λ_q has width equal to W , and is available in unlimited quantity. Thereby, a cutting pattern of one single stock length can be represented as $a^{jk} = (a_{1jk}, \dots, a_{|\bar{I}|jk})^T \in \{0,1\}^{|\bar{I}|}$ with $\sum_{i \in \bar{I}} w_i a_{ijk} \leq \lambda_k$. The relaxed model of 1CSPM is:

$$z^{qCSP} = \sum_j x_{jq} \longrightarrow \min, \text{ subject to} \quad (9)$$

$$\sum_k \sum_j a_{ijk} x_{jk} \geq h_i, \quad i \in \bar{I} \quad (10)$$

$$\sum_j x_{jk} \leq u_k, \quad k = \overline{1, q-1} \quad (11)$$

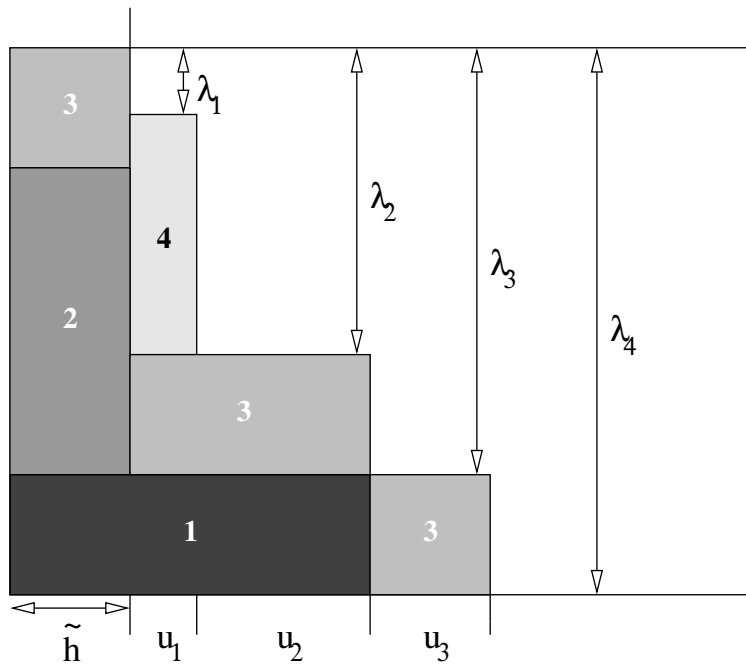


Figure 2: An example of a partial solution.

$$x_{jk} \in \mathbb{R}_+, \forall j, k \quad (12)$$

For the solution of the latter the column generation method with binary knapsack problem is used, cf. [4]. The number $\tilde{h} + \sum_{k=1}^{q-1} u_k + \lceil z^{qCSP} \rceil$ is used as a lower bound for the residual problem \bar{I} .

The approach is based on permutations of the item sequence and can produce many symmetrical solutions. In the first place there are many vertical and horizontal symmetries. To eliminate them we propose dominance criteria. We propose also a preprocessing procedure which is executed before branching. The procedure lets us find a subset of items which can be cut in only one optimal way.

Several strategies of search are considered. We combine best bound search and a diving procedure. In problems in which the optimum can not be found in acceptable time we use best bound search in order to improve the lower bound. This approach is different to [1] where depth-first search was used.

Results of computational experiments are shown on test instances from the literature, e.g., on the problems of Martello & Vigo and Berkley & Wang [3].

References

- [1] S. Martello, M. Monachi, D. Vigo, *An exact approach to the strip-packing problem*, INFORMS Journal on Computing 15(3): 310-319, 2003.
- [2] M. Monachi, *Algorithms for packing and scheduling problems*. PhD Thesis, Università di Bologna, 2001.
- [3] A. Lodi, S. Martello, D. Vigo, *Recent advance on two-dimensional bin packing problems*, Discrete Applied Mathematics 123: 379-396, 2002.

- [4] A. Caprara, M. Monaci, *Bidimensional packing by bilinear programming*, Mathematical Programming. Ser. A. Published online 30 August 2007.

Jürgen Rietz

Universidade do Minho
4710-057 Braga, Portugal

Special difficulties in the three-dimensional container loading problem

Problem formulation

Given a rectangular container of dimensions $L \times W \times H$, pack rectangular boxes of sizes $\ell_i \times w_i \times h_i$ into the container, such that no box contains points outside the container or penetrates another box. We assume that the boxes must not be rotated and their sides shall be parallel to the container sides. This problem is generally very difficult because of a lot of possible partial packing patterns.

A negative conjecture

If all boxes are packed one by one such that every box is moved as far as possible to the bottom left front (until it touches in all three directions another piece or the container) then probably there are instances of the packing problem, where no feasible packing of all boxes can be found in spite of its existence.

For motivation look at the figure 3. It shows, how five unit cubes and six boxes of sizes $1 \times 2 \times 4$ and $2 \times 2 \times 3$, which may be rotated, can be put into a cube of side length 5. In the corners each times three larger boxes come together and they cannot be moved to the corner, even if the unit cubes are removed. The entire packing is now mirrored several times and changed further. The result (without unit cubes) is shown in figure 4. For more flexibility choose a

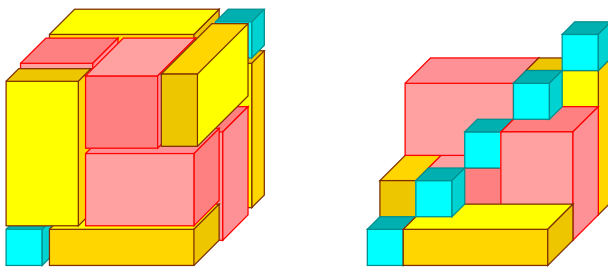


Figure 3: A three-dimensional puzzle

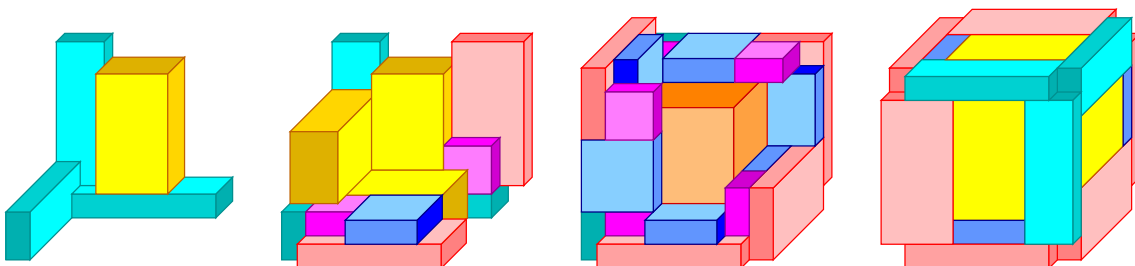


Figure 4: three-dimensional puzzle for the conjecture (here with $n = 4$)

parameter $n \in \mathbf{N}$ with $n \geq 3$. The side length of the large cube is $n + 4$. Ten unit cubes were removed. The other 31 boxes are arranged as in the table 5 given.

Table 5: solution of the puzzle

type	x	y	z	ℓ_i	w_i	h_i							
4	1	0	0	$n+2$	2	1	3	3	$n+1$	1	$n-1$	3	1
4	0	0	1	2	1	$n+2$	1	2	2	2	n	n	n
4	0	1	0	1	$n+2$	2	3	2	0	$n+2$	$n-1$	3	1
6	2	0	1	$n-1$	2	$n+1$	2	$n+2$	$n+1$	1	1	2	2
2	1	1	$n+1$	1	2	2	5	$n+3$	1	0	1	$n+2$	3
5	$n+1$	0	1	3	1	$n+2$	6	$n+2$	2	3	2	$n+1$	$n-1$
6	1	2	0	$n+1$	$n-1$	2	2	$n+1$	1	$n+2$	2	2	1
2	$n+1$	1	1	2	1	2	5	1	0	$n+3$	$n+2$	3	1
3	$n+2$	2	0	1	$n-1$	3	6	2	3	$n+2$	$n+1$	$n-1$	2
5	1	$n+1$	0	$n+2$	3	1	2	1	$n+2$	$n+1$	2	1	2
2	1	$n+1$	1	2	2	1	5	0	$n+3$	1	3	1	$n+2$
6	0	1	2	2	$n+1$	$n-1$	6	3	$n+2$	2	$n-1$	2	$n+1$
5	0	1	$n+1$	1	$n+2$	3	4	$n+2$	$n+3$	1	2	1	$n+2$
3	1	3	$n+1$	1	$n-1$	3	4	1	$n+2$	$n+3$	$n+2$	2	1
3	$n+1$	1	3	3	1	$n-1$	4	$n+3$	1	$n+2$	1	$n+2$	2

Trying to prove the conjecture

The packing problem can be modeled as an integer linear problem. The models of the BEASLEY type have the disadvantage of a large number of binary variables, especially when the container size is increased. Another model is due to FASANO and PADBERG, see e.g. [1]. As reference point always the lower left front corner is chosen.

Each piece must fit into the container. This yields the conditions

$$0 \leq x_i \leq L - \ell_i \quad (13)$$

$$0 \leq y_i \leq W - w_i \quad (14)$$

$$0 \leq z_i \leq H - h_i \quad (15)$$

Since no piece may overlap another, for any different indices i, j at least one of six inequalities must hold. This can be expressed by binary variables $X_{ij}, X_{ji}, Y_{ij}, Y_{ji}, Z_{ij}, Z_{ji}$, which tell, which of the inequalities is valid. Hence, the following conditions arise:

$$x_i - x_j + L * X_{ij} \leq L - \ell_i \quad (16)$$

$$x_j - x_i + L * X_{ji} \leq L - \ell_j \quad (17)$$

$$y_i - y_j + W * Y_{ij} \leq W - w_i \quad (18)$$

$$y_j - y_i + W * Y_{ji} \leq W - w_j \quad (19)$$

$$z_i - z_j + H * Z_{ij} \leq H - h_i \quad (20)$$

$$z_j - z_i + H * Z_{ji} \leq H - h_j \quad (21)$$

$$X_{ij} + X_{ji} + Y_{ij} + Y_{ji} + Z_{ij} + Z_{ji} = 1 \quad (22)$$

The conditions (13)–(22) yield an integer linear problem. Since 31 pieces shall be packed, 93 coordinates and $31 * 30 * 3 = 2790$ binary variables arise. The equation (22) can be dissolved,

such that one of the six binary variables can be eliminated. Since setting more than one of these six binaries to 1 cannot give more freedom in the conditions (16)–(21), it is no problem that removing one of the binary variables yields an extended feasible area. Because 15 of the 31 pieces are exactly the same as 15 other pieces, we can demand $x_i \leq x_{i+15}$ if the pieces are numbered appropriately ($i = 1, \dots, 15$). Then again 15 binary variables can be removed. But in spite of some further additional conditions, e.g. to remove some symmetry, the decision problem, if a solution exists, where one piece is put into the corner $(0,0,0)$, remains too complicated for ILOG CPLEX interactive optimizer.

References

- [1] PADBERG, M.: Packing small boxes into a big box. Math. Methods of Operations Research (ZOR), 52:121, 2000.

LP-based branching in the interval graph algorithm for orthogonal packing

Consider the d -dimensional orthogonal packing problem (OPP- d): given a set of orthogonal objects, n items and a container, it is to answer if the items can be packed in the container. No rotation is allowed. Fekete et al proposed to represent a feasible packing by a system of intersection graphs. Moreover, they defined 3 sufficient properties for an arbitrary system of d graphs to represent a valid packing: each graph should be interval (P1); each stable set of the k -th graph should have the sum of its weights not greater than the container size in dimension k (P2); the intersection of the graphs should be empty (P3). Fekete et al (2007) proposed a combinatorial enumeration scheme to construct such a system or to state that none exists. They used the so-called dual-feasible functions for bounds. This algorithm seems to be the most effective exact algorithm today.

We propose a Branch-&-Bound algorithm calculating in each node the d one-dimensional bar relaxations using LP with column generation. It is then checked if the obtained graphs represent a packing class. P2 is satisfied automatically. We discuss the main concepts of the branching strategy. Belov (2007) showed that P1 can be reduced to property P1': each graph is a cocomparability graph. We compare both variants and provide numerical results.

References

- [1] G.Belov and G.Scheithauer. Extending Rectangular Packing Classes to Cocomparability Graphs. 22nd European Conference on Operational Research EURO XXII, Prague 2007.
- [2] S. P. Fekete, J. Schepers, and J. C. van der Veen. An Exact Algorithm for Higher-Dimensional Orthogonal Packing, *Operations Research* 55(3), 2007, pp. 569-587.

Covering of a polygonal region by rectangles

The following 2D covering problem is considered: Let be given a collection of rectangles and a target compact polygonal region. Decide whether there exist translation vectors for the rectangles such that the union of the translated rectangles cover the target polygonal region, or not

In our investigations the covering rectangles can be different from each other, and the target polygonal region can be a nonconvex multi-connected compact object.

We construct a mathematical model of the covering problem based on a special function. This function is a mathematical tool for the analytical description of relationships between the collection of translated covering rectangles and the target polygonal region. In so doing we formalize a cover criterion using the Φ -function technique, i.e. if the target polygonal region is completely contained within the union of translated rectangles then the constructed Φ -function for the region and the complement of the union of translated rectangles has to have a nonnegative value.

This covering problem is multi-extremal and NP-complete. Therefore, we apply a modification of an exact branch-and-bound algorithm. We realize the search for a covering by constructing a search tree and by using of appropriate termination rules. The search ends either with translation vectors for the covering rectangles, i.e. with a covering, or with the result that there does not exist a cover, or with the statement that we can not find a cover.

Computational tests show that the running time substantially depends on the number of rectangles.

Guntram Scheithauer

Institut für Numerische Mathematik
Technische Universität Dresden

Zuschnitt- und Packungsoptimierung

A new book on cutting and packing optimization

A new book on cutting and packing optimization is presented. The contents of the book are as follows:

1. Modelling
2. Knapsack Problem
3. Cutting Stock and Bin Packing Problem
4. Optimal Guillotine Cutting
5. Optimal Allocation of Rectangles
6. Strip Packing of Rectangles
7. Quality Restrictions
8. Pallet Loading Problem
9. Container Loading Problem
10. Packing of Polygons
11. Circle and Ball Packing

Pradyumn Kumar Shukla

Institut für Numerische Mathematik
Technische Universität Dresden

An efficient, adaptive scalarization scheme for multi-objective discrete optimization

In this work we are interested in finding a representative sample of Pareto-optimal solutions of a multi-objective optimization problem. It has been widely emphasized that convergence and diversity are two conflicting criteria which are desired in any algorithm which tries to generate the entire efficient front. There are two broad classes of methods that are able to generate a sample of Pareto-optimal points. In the first class, many classical generating multi-objective optimization methods use an iterative scalarization scheme of standard procedures such as weighted-sum or ε -constraint method. The drawback of most of these approaches is that although there are results for convergence, diversity is hard to maintain. Thus we see that systematic variation of parameters in these scalarization techniques do not guarantee diversity in the solution set. The second class of methods use a scalarizing scheme that gives a good diversity of solutions. It starts from equidistant points on the utopia plane (plane passing through individual function minimizers) and then goes along a certain direction. However, a serious drawback of most of the methods of this class is that convergence is not guaranteed.

The contributions of this work are two-fold. First, we present a new algorithm guaranteeing both convergence and diversity. This improvement makes these algorithms theoretically equivalent to other classical algorithms (like weighted-sum or ε -constraint methods), without losing its ability to generate a set of evenly spaced efficient solutions. Second, we present an algorithm so as to know beforehand about certain sub-problems whose solutions are not Pareto-optimal and thus not wasting computational effort to solve them. This is useful for non-convex problems.

We present two real world discrete multi-objective examples that show the efficacy of the proposed algorithm in solving practical problems. The first is a mechanical gear train design problem and the second is a truss design optimization problem.

Schedule execution to minimize makespan for two-machine flow-shop with uncertain processing times

We address the issue of how to best execute the schedule in a two-phase scheduling decision framework by considering an uncertain two-machine flow-shop scheduling problem $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{max}$ in which each uncertain processing time $p_{ij} \in R_+$ of job $J_i \in \mathcal{J} = \{J_1, J_2, \dots, J_n\}$ by machine $M_j \in \mathcal{M} = \{M_1, M_2\}$ may take any real value between given lower bound $p_{ij}^L > 0$ and given upper bound $p_{ij}^U \geq p_{ij}^L$. The scheduling objective is to minimize the makespan C_{max} . There are two phases in the scheduling process under consideration: the off-line phase (the schedule planning phase) and the on-line phase (the schedule execution phase). The information of the lower bound p_{ij}^L and upper bound p_{ij}^U for each uncertain processing time p_{ij} is available at the beginning of the off-line phase. The local information on the realization p_{ij}^* (the actual value) of each uncertain processing time p_{ij} is available once the corresponding operation (of a job on a machine) is completed: $p_{ij} = p_{ij}^*$.

Let T denote the set of all possible vectors $p = (p_{1,1}, p_{1,2}, \dots, p_{n1}, p_{n2})$ of the uncertain processing times: $T = \{p \mid p \in R_+^n, p_{ij}^L \leq p_{ij} \leq p_{ij}^U, J_i \in \mathcal{J}, M_j \in \mathcal{M}\}$. In the off-line phase, a scheduler prepares a minimal set $S(T)$ of dominant permutations [1], which is derived based on a set of sufficient conditions for schedule domination that have been proven in [2; 3]. This set $S(T)$ of dominant permutations enables a scheduler to quickly make an on-line scheduling decision whenever additional local information on a realization of an uncertain processing time is available. Set $S(T)$ also can optimally cover all feasible realizations of the uncertain processing times in the sense that for any feasible realizations p_{ij}^* of the uncertain processing times p_{ij} there exists at least one permutation in set $S(T)$ which is optimal. Our approach enables a scheduler to best execute a schedule and may end up with executing the schedule optimally in many instances according to our extensive computational experiments which were based on randomly generated data up to $n \leq 1000$ jobs. The algorithms for testing the set of sufficient conditions of schedule domination are not only theoretically appealing (i.e., polynomial in the number n of jobs) but also empirically fast as our extensive computational experiments indicate.

We would like to mention that there is another scheduling research line in dealing with uncertain processing times, e.g., one with a decision criterion of minimizing the worst-case regret [4; 5; 6]. Basically, the latter scheduling research line deals with the off-line phase only. In this research line, one aims to seek one schedule that is optimal from a decision criterion (i.e., minimizing the worst-case regret) and no attempts are made to take advantage of the local on-line information to best execute the schedule as the scheduled process goes on. As a new scheme dealing with uncertainty, our two-phase scheduling scheme must be tested on a representative class of uncertain scheduling problems. To this end, two-phase scheduling algorithms were coded in C++ and were tested on a large number of randomly generated problems $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{max}$. The computational results obtained in our experiments seems to be rather promising especially for on-line scheduling phase.

In particular, the off-line scheduling allowed us to find optimal schedules for small numbers of jobs ($n \leq 40$) and small errors of input data. For $n > 40$ there were no such randomly generated instances at all. Fortunately, on-line scheduling allowed us to find optimal schedules (with optimality proofs before schedule execution) for most randomly generated instances with small and medium n , $n \leq 100$, and for many randomly generated instances with large n , $200 \leq$

$n \leq 1000$. The value C_{max} obtained for the actual schedule turns out to be equal to the optimal value C_{max}^* calculated for optimal schedule with the actual job processing times p_{ij}^* . It should be noted that value C_{max}^* can be calculated after completing the last job from set \mathcal{J} when all actual job processing times $p_{ij}^* \in T$, $J_i \in \mathcal{J}$, $M_j \in \mathcal{M}$, and all actual job completion times become known. The following computational results obtained in our experiments were even more impressive. The average relative error of the makespan $[(C_{max} - C_{max}^*)/C_{max}^*] \cdot 100\%$ obtained for all actual schedules was less than 2.9% for all randomly generated instances with small number of jobs: $n \leq 10$. The average relative error of the makespan obtained for all actual schedules was less than 1.67% for all randomly generated instances with n jobs with $20 \leq n \leq 1000$. These results were obtained since the percentage of the correct decisions made in on-line scheduling phase was rather high. Thus, the sufficient conditions for the existence of a dominant permutation [2; 3] may be very effective for on-line scheduling. It should be also noted that the number of decision-making time-points when the order of conflicting jobs has to be decided was rather high for some instances with $n \geq 50$. However, these decisions made in our algorithms were very fast: there were no randomly generated instance which takes a running time more than 0.05 seconds for a processor with 1200 MHz.

References

- [1] T.-C. Lai, Yu.N. Sotskov, N. Sotskova and F. Werner, Optimal makespan scheduling with given bounds of processing times, *Mathematical and Computer Modelling*, **26**, 67–86, 1997.
- [2] N.M. Leshchenko and Yu.N. Sotskov, Two-machine minimum-length shop-scheduling problems with uncertain processing times, In *Proceedings of XI-th International Conference "Knowledge-Dialogue-Solution"*, Varna, Bulgaria, June 20-30: Vol. 2, 375–381, 2005.
- [3] N.M. Leshchenko and Yu.N. Sotskov, Realization of an optimal schedule for the two-machine flow-shop with interval job processing times, *International Journal "Information Theories and Applications"*, **14** (2), 182–189, 2007.
- [4] S.D. Wu, E.-S. Byeon and R. Storer, A graph-theoretical decomposition of the job-shop scheduling problem to achieve scheduling robustness, *Operations Research*, **47** (1), 113–124, 1999.
- [5] P. Kouvelis, R.L. Daniels and G. Vairaktarakis, Robust scheduling of a two-machine flow shop with uncertain processing times, *IIE Transactions*, **32**, 421–432, 2000.
- [6] V. Lebedev and I. Averbakh, Complexity of minimizing the total flow time with interval data and minmax regret criterion, *Discrete Applied Mathematics*, **154**, 2167–2177, 2006.

Tatiana Starostina

Zurich University of Applied Sciences, Life Sciences and Facility Management
Institut of Applied Simulation

Solving dynamic flow problem using simulation

A simulation is an imitation of the operation of a real-world process or system over the time [1]. Simulation is used to describe and analyse behaviour of a system and to optimize a work of the system. As such a system a transportation network can be considered.

A transportation network G is described by directed graph $G = (V, A)$ consisting of the node set V with a source $s \in V$, a sink (destination) $t \in V$ and the arc set A . Each arc a_{ij} from the set A is characterized by capacity c_{ij} and transit time τ_{ij} on the arc. In a feasible dynamic flow at most f_{ij} units of flow can be transported along arc a_{ij} with each time step. Flow leaving a node v_i at time t reaches node v_j at time moment $t + \tau_{ij}$.

There are some statements of dynamic flow problem in transportation network: maximum dynamic flow problem and quickest dynamic flow problem. Each problem considers a network with one search and one destination. A maximum flow sends from the search to the destination as much flow as possible within a defined time period. A quickest flow sends a specified amount of flow from the search to the destination in the shortest possible time [2]. As it has been shown in [3], the maximum dynamic flow problem can be solved using minimum cost flow algorithm. The quickest dynamic flow problem can be reduced to the maximum dynamic flow problem by binary search.

Instead replacement the dynamic problem with a static equivalent and use static optimisation algorithm, we can consider this problem dynamic directly, using simulation.

We investigate discrete dynamic network flow problem using discrete event simulation approach. This approach begins of construction of a simulation model. Discrete simulation model describes a transportation network with all parameters and considers many state changes. The state variables can change their values only at certain time points and the model states are separated clearly from each other [1]. Discrete event simulation model is defined as one in which the state variables change only at those discrete points in time at which events occur [1]. After construction of simulation model the various experiment can be fulfilled on this model and optimal solution can be found.

A simulation approach solves the same problems as an optimization one, but by another way. Simulation can be very useful especially for solving dynamic problems, also dynamic flow problems.

References

- [1] J. Banks, *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, J.Wiley and sons: Engineering and Management Press, New York, 1998.
- [2] B. Hoppe, E. Tardos, *Polynomial Time Algorithms for Some Evacuation Problems*, Proceedings of the fifth annual ACM-SIAM symposium on discrete algorithms, Arlington, Virginia, 1994, p. 433-441.
- [3] L.R. Ford, D.R. Fulkerson, *Flows in Networks*, Princeton University Press, New Jersey, 1962.

Rico Walter

Faculty of Mathematics and Computer Science
Friedrich-Schiller-University Jena

Competitive Scheduling Scenarios

Scheduling as well as game theory are two classical areas in applied mathematics of great academic interest and broad practical relevance. Till now, nearly all studies on these two areas have been done independently from each other and only a few game-theoretic approaches to scheduling are known. Therefore the goal of my work is to bridge the gap between scheduling and combinatorial game theory.

For it two rather different scenarios that are more or less closely related to this issue will be presented. At first I will introduce a generalized class of subtraction games and secondly I will deal with load balancing scenarios in presence of a random decision maker.

The first - rather abstract - scheduling scenario is a generalization of the well-known two-person game called *subtraction game*. In our modified variant we keep to the classic situation but additionally consider local payoffs for every move. We assume cyclic, integer payoffs and call this game class *subtraction games with cyclic local payoffs*. In a zero-sum sense the two players try to maximize the difference between their total payoff, which is the sum of all of their received local payoffs, and the opposing total payoff. This difference is called game-theoretic value.

A question of great interest is, for which cycle lengths and sets of legal moves the sequence of the game-theoretic values becomes periodic. My work mainly concentrates on games with cycle length two and I will give a total characterization concerning periodicity for it.

The basic situation of the second scheduling scenario is the same as in load balancing problems. Given are a finite set of m identical machines and a finite set of n jobs with positive processing times that have to be processed by any of the machines. The objective of the scheduler is to find an assignment of all jobs to the machines that minimizes the maximum load (also called makespan) over all machines, where the load of a machine is defined by the sum of the processing times of the jobs that are assigned to this machine. In case $m = 2$ this problem is identical to the partition problem and thus NP-complete.

By adding a random decision maker we transfer the basic, deterministic situation into a situation that can be interpreted as “a game against nature” and therefore might containing some competitive aspects. The task of the scheduler is the same as above, but now he has additionally to cope with the random decisions during the assignment.

To tackle this problem we use heuristic approaches based on greedy-strategies with different evaluation functions that sound promising for the deterministic case. Experimental studies show that on average such simple strategies still yield pretty good results in presence of a random decision maker.

It's not that surprising that partly random decisions are sometimes capable of generating better solutions or even an globally optimal one. But what is more surprising is the fact, that instances exist, for which the objective function value gained by the deterministic version is an upper bound for the worst case performance of the randomized version. So to sum up, we should preferably not talk about “a game against nature” in general, because of the benefit that can be achieved in some cases.

18th Workshop on Discrete Optimization

Königstein, May 18–21, 2008

List of Participants

- **Prof. Althöfer, Ingo**, Faculty of Mathematics and Computer Science, Friedrich-Schiller University Jena (p. 6) `althofer{at}mathematik.uni-jena.de`
- **Andresen, Michael**, Otto-von-Guericke Universität Magdeburg (p. 7) `Michael.Andresen{at}Mathematik.Uni-Magdeburg.DE`
- **Dr. Belov, Gleb**, Technische Universität Dresden (p. 7, 19, 25) `Gleb.Belov{at}tu-dresden.de`
- **Prof. Bräsel, Heidemarie**, Otto-von-Guericke Universität Magdeburg (p. 8) `heidemarie.braesel{at}math.uni-magdeburg.de`
- **Prof. Dempe, Stefan**, TU Freiberg `dempe{at}tu-freiberg.de`
- **Dörnfelder, Martin**, Faculty of Mathematics and Computer Science, Friedrich-Schiller-University Jena (p. 9) `martin_doernfelder{at}gmx.de`
- **Dr. Fanghänel, Diana**, Institut für Informatik, Universität zu Köln (p. 10) `fanghaenel{at}informatik.Uni-Koeln.DE`
- **Prof. Fischer, Andreas**, Technische Universität Dresden `Andreas.Fischer{at}tu-dresden.de`
- **Prof. Girlich, Eberhard**, Otto-von-Guericke Universität Magdeburg `eberhard.girlich{at}mathematik.uni-magdeburg.de`
- **Hemig, Claas**, Clausthal University of Technology (p. 12) `claas.hemig{at}tu-clausthal.de`
- **Prof. Hempel, Lorenz**
- **Mesyagutov, Marat**, Technische Universität Dresden (p. 19) `mmesyagutov{at}gmail.com`
- **Dr. Rietz, Jürgen**, Universidade do Minho (p. 22) `Juergen_Rietz{at}gmx.de`
- **Rohling, Heide**, Technische Universität Dresden (p. 25)
- **Prof. Romanova, Tatjana**, Institute for Mechanical Engineering of the National Academy of Sciences of Ukraine (p. 26) `sherom{at}kharkov.ua`
- **Dr. Scheithauer, Guntram**, Technische Universität Dresden (pp. 19, 26) `Guntram.Scheithauer{at}tu-dresden.de`

- **Shukla, Pradyumn K.**, Technische Universität Dresden (p. 28)
pradyumnshukla{at}yahoo.com
- **Prof. Sotskov, Yuriy**, United Institute of Informatics Problems, Minsk (p. 29)
sotskov{at}newman.bas-net.by
- **Dr. Starostina, Tatjana**, Zurich University of Applied Sciences, Life Sciences and Facility Management Institut of Applied Simulation (p. 31)
stat{at}zhaw.ch
- **Prof. Stoyan, Yuriy**, Institute for Mechanical Engineering of the National Academy of Sciences of Ukraine (p. 26)
- **Walter, Rico**, Faculty of Mathematics and Computer Science, Friedrich-Schiller-University Jena (p. 32) ricow{at}minet.uni-jena.de