Dagstuhl Seminar No. 0031

Constraint Programming and Integer Programming

organized by

Krzysztof Apt (Amsterdam)
Michael Jünger (Köln)
Pascal van Hentenryck (Providence)
Laurence A. Wolsey (Louvain-La-Neuve)

Contents

1 Preface

In the **Mathematical Programming Community** (consisting mainly of mathematicians and operations researchers) **Integer Programming (IP)** started flourishing in the 1950's when cutting plane and branch&bound algorithms were proposed for the solution of

$$\text{(IPP)} \qquad \begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \text{ integer} \end{array}$$

where c and b are n- and m-dimensional vectors, respectively, and A is an m by n matrix of rational numbers. Refinements and hybrids of both techniques led to powerful algorithms for IPP as well as combinatorial optimization problems with linear objective function

$$\text{(COP)} \qquad \begin{array}{ll} \text{maximize} & \sum_{e \in F} c_e \\ \text{subject to} & F \in \mathcal{F} \end{array}$$

where for a finite set E a collection of subsets $\mathcal{F} \subseteq 2^E$ defines the feasible solutions. Prominent successes are, e.g., the most powerful software packages for the traveling salesman problem.

In the **Artificial Intelligence Community** (consisting mainly of logicians and computer scientists) **Constraint Programming (CP)** started flourishing in the 1980's after constraint satisfaction problems had been a focus of research in the 1970's. Today's state of the art includes several systems that support programming language constructs whose expressive power goes far beyond linear equations and inequalities to include logical and higher-order constraints, global constraints, as well as specific support for scheduling and resource allocation problems. These languages also let users to specify and control a search procedure appropriate for a given application. Prominent successes are, e.g., in the area of production scheduling.

Many areas of applications of constraint programming techniques are beyond the scope of integer programming techniques, but Integer Programming and Constraint Programming are complementary techniques for combinatorial optimization problems of the form COP. Depending on the problem, one or the other technique is more appropriate.

Software for general integer programming, and special purpose software for structured problems such as the traveling salesman problem or the max cut problem, almost always depends on the formulation (IPP). Common features of the algorithms are a priori preprocessing, enumeration in which the bounds provided by linear programming play a crucial role in pruning the enumeration tree, and cutting planes that are used to tighten the linear programming bounds, and push the linear programming solution closer to integrality.

Using an IPP formulation is an advantage in terms of generality, but as concerns flexibility,

i) modelling certain simple situations/expressions may require a very large number of variables,

ii) the preprocessor must maintain an IPP formulation, and therefore only a restricted set of transformations can be carried out,

iii) certain IPP formulations have very weak linear programming relaxations so the bounds provided are useless, and

iv) the choice of how to branch is also restricted by the IPP formulation.

CP is also based on enumeration, but typically has a richer language to represent a problem, allowing for more refined preprocessing and branching. The basic idea behind CP systems is to enforce some consistency notions on the constraints (e.g., arc-consistency) at each node of the search tree. These consistency notions define in fact relaxations of the problem. In general, these relaxations are specific to classes of applications (e.g. edge finder in scheduling) and are used to reduce the set of possible values that the decision variables can take. In so doing, they also produce lower bounds for optimization problems. However, CP systems have not exploited so far the wealth of results provided by linear programming relaxations and their associated cuts.

Only very recently, first scientific links between IP and CP were installed, but still, both subjects develop independently to a large extent. Reasons include the different scientific contexts in which the two disciplines developed. This results in language barriers. Integer programmers use mathematical concepts that are unfamiliar to constraint programmers who in turn use computer science concepts that are unfamiliar to integer programmers.

After the seminar, the organizers are even more convinced that combinations of IP and CP techniques will have an increasing impact on combinatorial problem solving, and that the potential of joint efforts is largely unexplored. On Monday and Tuesday morning we tried to overcome the language barriers by asking prominent researchers to give introductions for the other community into the most relevant topics. This first part required a thorough preparation of the lectures and was be planned in advance (which is unusual for a Dagstuhl seminar). From Tuesday afternoon on, the seminar followed the usual scheme: The talks were scheduled "on-line" according to need, and gradually, more and more interaction between the participants of the two communities developed. On Thursday evening, we had a panel discussion with two panelists of either community in which the situation was reflected and it was tried to identify problems for which a fruitful interaction/competition of constraint programming techniques and integer/combinatorial optimization techniques appears challenging and is likely to enhance the interaction of both communities. We are confident that the seminar was an important step towards this goal.

Amsterdam, Köln, Providence, Louvain-la-Neuve, April 10, 2000

Krzysztof Apt, Michael Jünger, Pascal van Hentenryck, Laurence Wolsey

2 Final Program

Monday, 17 January 2000

07:30	–	**08:45**	Breakfast
08:45	–	**09:00**	Introduction by the Organizers
09:00	–	**10:30**	Pascal van Hentenryck *The OPL Optimization Programming Language*
10:30	–	**10:45**	Coffee
10:45	–	**12:15**	George Nemhauser / Michael Trick *Integer Programming Modeling*
12:15			Lunch
14:00	–	**15:30**	Toby Walsh *Non-binary Constraints*
15:30	–	**16:00**	Coffee
16:00	–	**17:30**	Gerhard Reinelt *Introduction to Combinatorial Optimization*
18:00			Dinner

Tuesday, 18 January 2000

07:30	–	**09:00**	Breakfast
09:00	–	**10:30**	Jean-Charles Régin / Barbara Smith *Constraint Programming: Models and Algorithms*
10:30	–	**10:45**	Coffee
10:45	–	**12:15**	Laurence A. Wolsey *Introduction to Integer Programming*
12:15			Lunch
16:00	–	**16:30**	Mark Wallace *The ECLIPSE Language for Hybridizing CP and IP*
16:30	–	**17:15**	Philipe Refalo *Linear Formulations of Constraint Programming Models and Hybrid Solvers*
17:15	–	**18:00**	Alexander Bockmayr *Branch-and-Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming*
18:00			Dinner

Wednesday, 19 January 2000

07:30 – **09:00** Breakfast

09:00 – **09:45** John Hooker
Integrating IP and CP

09:45 – **10:15** Fritz Eisenbrand
Cutting Planes and Cutting Plane Closure in Fixed Dimension

10:15 Coffee

10:30 – **11:15** Rina Dechter
On the Automatic Generation of Lower Bound Functions for Branch & Bound

11:15 – **12:00** Nicolas Beldiceanu
Global Constraints as Graph Properties on Structured Networks

12:15 Lunch

13:45 Excursion to Trier

Thursday, 20 January 2000

07:30	–	**09:00**	Breakfast
09:00	–	**09:30**	Karen Aardal *Lattice Approaches to Integer Programming*
09:30	–	**10:00**	Jan Karel Lenstra *WHIZZKIDS - Two Exercises in Computational Discrete Optimization*
10:00			Coffee
10:30	–	**11:00**	Alexander Martin *Structure Analysis of Integer Programs*
11:00	–	**11:30**	Michaela Milano *Cost-based Domain Reduction: An Application to the Travelling Salesman Problem with Time Windows*
11:30	–	**12:00**	Martin Savelsbergh *Preprocessing and Probing*
12:15			Lunch
14:00	–	**14:30**	Egon Balas *Tight Representation of Logical Constraints*
14:30	–	**15:00**	Rolf Möhring *LP-guided Search for good Solutions*
15:00	–	**15:30**	Susanne Heipcke *An Approach to Combined Modeling and Solving in CP and MIP*
16:00	–	**18:00**	Panel Discussion Panelists: Nicolas Beldiceanu, George Nemhauser, William Pulleyblank, Pascal van Hentenryck
18:00			Dinner

Friday, 21 January 2000

07:30 – 09:00 Breakfast

09:00 – 09:30 Kurt Mehlhorn
Bound Consistency of the Sortedness Constraint

09:30 – 10:00 William Pulleyblank
Hilbert Bases and Cutting Planes

10:00 Coffee

10:15 – 10:45 Laurent Michel
A Modeling Language for Local Search

10:45 – 11:15 Robert E. Bixby
A Paradigm for Finding Cuts

11:15 – 11:45 Mike Trick
Integer Programming Models in Sports Scheduling

12:15 Lunch

3 Abstract of Presentations

Pascal van Hentenryck

The OPL Optimization Programming Language

OPL is a modeling language for constraint and mathematical programming. The purpose of the tutorial is to introduce constraint programming, both from a language and solver viewpoint, and to contrast it with mathematical programming. Applications in sport scheduling, resource scheduling and lot sizing were used to illustrate the two technologies.

George Nemhauser / Michael Trick

Integer Programming Modeling

In the last decade, the use of mixed integer programming (MIP) has increased dramatically because it is now possible to solve problems with thousands of integer variables on a personal computer and to obtain provably good solutions to even much larger problems. These advances have been made possible by developments in modeling, algorithms, software and hardware. This talk focuses on effective modeling, preprocessing, and the methodologies of branch-and-cut and branch-and-price, which are the techniques that make it possible to treat models with either a very large number of constraints or variables.

Toby Walsh

Non-binary Constraints

One of the great strengths of constraint programming are specialized procedures for dealing with specific non-binary constraints like the all-different constraint. I survey recent work which helps us to understand the benefits of using such non-binary constraints. I also look at the impact of various encodings of non-binary constraints.

Gerhard Reinelt

Branch-and-Cut Algorithms for Combinatorial Optimization

Branch-and-Cut algorithms are a widely used tool for solving combinatorial optimization problems to optimality or to a prespecified quality guarantee. In this talk we survey this approach and discuss several aspects of its implementation. In particular we focus on the separation problem which is at the core of branch-and-cut. We address the question of how to gain knowledge about the facet structure of combinatorial polytopes and of how to design effective separation procedures using techniques like projecting and lifting or small facet separation. A short presentation of the software framework ABACUS for implementing branch-and-cut algorithms concludes the talk.

Jean-Charles Régin / Barbara Smith

Modeling for Constraint Programming - A Case Study

This talk discusses different ways of modeling the problem of finding a pair of orthogonal Latin squares as a constraint satisfaction problem and compares the performance of different models and search strategies. A pair of $n \times n$ Latin squares are orthogonal if the pairs of numbers in each row/column are all different. The problem can be modeled using 0-1 variables and linear constraints; however, a CP model is not restricted to linear constraints, but can instead use the 'all-different' constraint. The decisions to be made in devising a search strategy for a proposed CP model are discussed (e.g. whether to treat the all-different constraints globally, or as collections of binary not-equal constraints; which variables to use as search variables and in what order to assign them). It is also shown that an equivalent CP model can be derived. Combining the two equivalent models gives better constraint propagation and hence allows solutions to be found more quickly. Experimental comparisons of models and search strategies are given.

Laurence A. Wolsey

Introduction to Integer Programming

After introducing cuts for general mixed integer programs, including mixed integer rounding and Gomory mixed integer cuts, various cuts treating local "canonical" structure, such as cover and flow-cover inequalities are presented. Given separation routines for such families of inequalities, the importance of interface routines finding the appropriate canonical structures is emphasized, and the branch-and-cut system BC-OPT is used as an example.
The alternative decomposition approaches, opposing partial convexification with separation subproblems versus Lagrangian relaxation with optimization subproblems is discussed, and some possible parallels with global constraints are suggested.

Mark Wallace

The ECLiPSE Language for Hybridizing CP and IP

ECLiPSE is a constraint logic programming language which supports problem modeling and solving in a way that is highly expressive, flexible and easy to modify.
ECLiPSe builds on logic programming, adding interfaces to solvers such as XPRESS, CPLEX, and ILOG SOLVER, and libraries such as the finite domain solver, interval solver and repair library. The extensibility is supported by advanced, but simple-to-use, control facilities and attributed variables.
ECLiPSe has been used to solve both hard benchmarks and large industrial applications, employing hybrid algorithms which combine linear constraint solving and finite domain propagation. An example is "A Generic Model and Hybrid Algorithm for Hoist Scheduling Problems", Proc CP'98, also at

`http://www.icparc.ic.ac.uk/eclipse/reports/index.html`

ECLiPSe is free to researchers. Papers, code examples, and download facilities are at `www.icparc.ic.ac.uk/eclipse/`

Philipe Refalo

Linear Formulations of Constraint Programming Models and Hybrid Solvers

We propose a linear formulation of some constraint programming models including global constraints and piecewise linear functions. We also present a scheme for hybrid CP-IP solver were the linear formulation is automatically strengthened during the search by way of cutting planes. An example of this cooperation is given with piecewise linear optimization.

Alexander Bockmayr

Branch-and-Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming

Integer linear programming and finite domain constraint programming are two general approaches for solving hard combinatorial problems. We present a unifying framework, *branch-and-infer*, to clarify the relationship between these two approaches and to show how they can be integrated.
Branch-and-infer is based on a distinction between primitive and non-primitive constraints. Primitive constraints are those constraints that can be solved easily and for which global methods are available. Non-primitive constraints are those constraints for which such methods do not exist and which make the problem hard to solve. In integer linear programming, the primitive constraints are linear equations and inequalities, which are solved over the real (or rational) numbers. The only non-primitive constraint is `integer`, i.e. the condition that some or all variables should take integer values. In finite domain constraint programming, the primitive constraints are domain constraints of the form $x \leq 2, y \geq 3, z \neq 4, x = y$, which are solved over the integer numbers. All other constraints are non-primitive. This includes more general arithmetic constraints, like linear equations, inequalities or disequalities in several variables, and symbolic constraints like `alldifferent` or `cumulative`.
The basic idea underlying the branch-and-infer framework is that, in both integer linear programming and finite domain constraint programming, problems are solved by a combination of inference and search. The primitive constraints define a relaxation of the problem, for which an efficient global solution method is available. The non-primitive constraints are handled locally by an inference agent that derives from a given non-primitive constraint and the current relaxation new primitive constraints that tighten this relaxation. Since, in general, a problem cannot be solved using the relaxation alone, inference has to be combined with search, which together provide a complete solution method. In integer linear programming, the primitive constraints are solved by linear programming methods, e.g. the Simplex algorithm. To handle the non-primitive constraint `integer`, general cutting plane techniques, e.g the Gomory-Chvátal method or disjunctive programming, can be applied as inference algorithms. In finite domain constraint programming, the non-primitive constraints are handled

by local consistency algorithms that reduce the domain of the variables, which corresponds to the inference of new bound inequalities or disequalities in the branch-and-infer framework. Branch-and-infer not only clarifies the relationship between integer linear programming and finite domain constraint programming, it can also be used to combine the two approaches. In particular, branch-and-infer shows how to transfer the idea of symbolic constraints from constraint programming into integer programming.
Symbolic constraints in integer programming allow the modeller to include large families of linear inequalities into the model, without writing them down explicitly. For example, when solving a traveling salesman problem, we may use a symbolic constraint `tsp`(...) to state the problem-defining degree and the subtour elimination constraints. Declaratively, this constraint is equivalent to exponentially many linear inequalities. Operationally, however, only some of these inequalities will be added to the model at runtime (as cutting planes).
Concerning efficiency, symbolic constraints allow one to integrate specialized cutting plane algorithms based on polyhedral combinatorics into a general branch-and-cut solver. Symbolic constraints give the modeller the possibility to identify some specific structure in the problem, which later can be exploited when the model is solved. For example, when we solve a model containing the symbolic constraint `tsp`, we can enhance our general branch-and-cut solver by computing specialized cutting planes for `tsp` instead of using more general cutting planes for arbitrary linear 0-1 programs.

[1] A. Bockmayr and T. Kasper. Branch-and-infer: A unifying framework for integer and finite domain constraint programming. *INFORMS J. Computing*, 10(3):287 – 300, 1998.

John N. Hooker

Integrating Constraint Programming and Integer Programming

This talk surveys three rationales for the integration of CP and IP.

a) The global constraints of CP (which represent specially-structured subsets of constraints) can be associated with relaxations and Benders cuts as well as the filtering algorithms that characterize CP.

b) The relaxation technology of IP can be applied to global and other constraints, and generalized via relaxation duality.

c) The inference methods of CP can strengthen IP and provide a framework for understanding many presolve techniques in IP. Substructures of a problem can give rise to new discrete constraints, as they give rise to cutting planes.

Fritz Eisenbrand

Cutting Planes and Cutting Plane Closure in Fixed Dimension

The elementary closure P' of a polyhedron P is the intersection of P with all its Gomory-Chvátal cutting planes. P' is a rational polyhedron provided that P is rational. The known bounds for the number of inequalities defining P' are exponential, even in fixed dimension. We show that the number of inequalities needed to describe the elementary closure of a rational polyhedron is polynomially bounded in fixed dimension. If P is a simplicial cone, we construct a polytope Q, whose integral elements correspond to cutting planes of P. The vertices of the integer hull Q_I include the facets of P'. A polynomial upper bound on their number can be obtained by applying a result of Cook et al. Finally, we present a polynomial algorithm in varying dimension, which computes cutting planes for a simplicial cone that correspond to vertices of Q_I.

Rina Dechter

On the Automatic Generation of Lower Bound Functions for Branch and Bound

We present a general new scheme that generates search heuristics for solving constraint optimization problems, mechanically. These heuristics, (lower bound functions for minimization task and upper bound for maximization), are extracted by a recently proposed approximation scheme called *mini-bucket* elimination that allows controlled tradeoff between computation and accuracy. The mini-bucket approach extends the principle of bounded inference known as local-consistency (e.g., arc-consistency) that proved so useful for constraint processing, to cost functions and to optimization. We show that the heuristic function generated can be used to guide Branch-and-Bound and Best-First search. The performance of the scheme is evaluated empirically on a number of optimization problems, including coding and medical diagnosis problems. Our results demonstrate that both search schemes are effective, permitting controlled tradeoff between preprocessing (for generating the lower bounding functions) and search.
The scheme can be viewed as a generalization of Branch and Bound for integer programming, to general constraint optimization, removing the restriction of linear constraints and linear objective function.

Nicolas Beldiceanu

Global Constraints as Graph Properties on Structured Networks

This talk introduces a classification scheme for global constraints. This classification is based on four basic ingredients from which one can generate almost all existing global constraints and come up with new interesting constraints. Global constraints are defined in a very concise way, in term of graph properties that have to hold, where the graph is a structured network of same elementary constraints. Since this classification is based on the internal structure of the global constraints it is also a strong hint for the pruning algorithms of the global constraints.

Karen Aardal

Lattice Approaches to Integer Programming

The theory of lattices and lattice bases has been used to derive several results in the theory of integer programming, the most prominent being that the integer programming problem can be solved in polynomial time if the dimension is fixed (H.W. Lenstra, Jr., 1983). In computational integer programming ideas from lattice theory have been used to a lesser extent. We describe two lattice approaches that have proved useful in recent computational work: branching on hyperplanes and problem reformulation. Computational results are provided indicating the power of these techniques on hard integer programming instances.

Jan Karel Lenstra

WHIZZKIDS – Two Exercises in Combinatorial Optimization

In 1996 and 1997 the Department of Mathematics and Computing Science at Eindhoven University of Technology organized two contests in cooperation with the software firm CMG Nederland and the newspaper De Telegraaf. The purpose of these contests was to increase interest in mathematics and computer science among highschool students. The participants had to construct a newspaper delivery scheme in 1996 and a timetable for a parents' evening at a high school in 1997. Both times they faced an optimization problem which was easy to formulate but hard to solve, and which caused exciting evenings and sleepless nights to both the puzzler at the kitchen table and the advanced algorithm designer.
I will discuss the background of the "Whizzkids contests" and describe how the tools of combinatorial optimization can be applied in finding good solutions and in attempting to prove that no better solutions exist. These tools include upper bounding techniques based on local search and lower bounding techniques using linear programming and constraint satisfaction.

Alexander Martin

Structure Analysis of Integer Programs

A mixed integer program (MIP) is to minimize a linear objective function subject to a set of linear constraints, where some or all of the variables must be integer. General MIP Solvers are faced with the problem that they have to extract all relevant information for cutting plane generation, branching strategies and so forth from the constraint matrix A, the right-hand side vector and the objective function. They do not know the application that led to this formulation. Thus, most solvers first preprocess the problem to tighten the formulation and to extract as much information as possible. Most preprocessing procedures, however, are just column or row based and do not look at the matrix as a whole. In this talk we analyze the structure of the matrix A and ask whether it has a certain form (so-called bordered block diagonal form) or whether it can be brought into this form by reordering columns and rows. We show that many MIP problems do indeed have bordered block diagonal form. We illustrate on some models that this fact can be exploited polyhedraly by deriving some new

cutting planes. Finally, we investigate whether bordered block diagonal form might help to derive new branching strategies.

Michaela Milano

Cost-based Domain Reduction: An Application to the Traveling Salesman Problem with Time Windows

I present a methodology for integrating optimization components in global constraints in order to perform cost based domain filtering in constraint programming. The idea is to provide the user with global constraints embedding an optimization component, representing a linear relaxation of the constraint itself, able to compute the optimal solution of the relaxation and a gradient function that provides the cost of each variable-value assignment. With these informations, we can perform pruning on the basis of costs, by removing values which cannot lead to solutions better that the best found so far.
I show an application which exploits this technique: the traveling salesman problem with time windows. I describe how to model it in constraint programming as the union of a TSP and a scheduling problem. I present some computational results that show that we outperform previous CP approaches and we are competitive with state of the art Branch and Cut approaches. Some future directions are also addressed.

Martin Savelsbergh

Preprocessing and Probing

We discuss various techniques that can be efficiently implemented to detect infeasibility, to detect redundancy, and to improve the bounds on variables. In term these techniques become more powerful when combined with probing. Probing refers to tentatevly setting a binary value to one of its bounds and investigating the consequences. Probing may lead to fixing variables, improving coefficients, and identification of logical implications. Logical Implications may be used as cutting planes themselves, or can be used in the constructing of conflict graphs, which allow derivation of stronger cutting planes.

Egon Balas

Tight Representation of Logical Constraints

Logical constraints involving linear inequalities can be viewed as unions of polyhedra. Optimization over unions of polyhedra has been studied since the '70-s under the heading of disjunctive programming, with the main focus on a compact representation of the convex hull. This higher dimensional representation is of size polynomial in the number of variables and constraints, and linear in the number of polyhedra in the union. It can be projected back onto the original space, where it typically gives rise to exponentially many inequalities. When applied to general mixed 0-1 programming, this approach is known as lift-and-project. We illustrate the technique by giving a convex hull representation of logical constraints known

as cardinality rules. For references, see the author's survey paper "Integer Programming: Introduction and Outline", distributed at the Dagstuhl conference.

Rolf Möhring

LP-guided Search for good Solutions

The construction of good feasible solutions from optimal solutions of a relaxation of the original problem is an important paradigm in integer linear programming.
We demonstrate this in the context of resource-constrained project scheduling.
Starting from a well-known time indexed ILP formulation (P) we consider the Lagrangian relaxation (LR) obtained by dualizing the resource constraints. The LP relaxation of (LR) is known to be integral. We show that (LR) can in fact be modeled as a min cut problem and thus be solved efficiently by a max flow algorithm.
We solve this min cut problem for different choices of Lagrangian multipliers within a subgradient optimization loop. Every such choice produces a schedule that may violate some resource constraints.
In order to turn these infeasible schedules into feasible ones, we use the technique of list scheduling by α-completion times, which has recently been very successful in the design of approximation algorithm for machine scheduling problems.
Computational experiments by Stork and Uetz on two classes of benchmark problems (PSPLIB, labor constrained scheduling problems) show that one obtains excellent schedules very fast. The quality of these schedules matches that of much more elaborate methods.
The lecture is based on joint work with Andreas Schulz (MIT), Frederik Stork (TU Berlin), and Mark Uetz (TU Berlin).

Susanne Heipcke

An Approach to Combined Modeling and Solving in Constraint Programming and Mixed Integer Programming

In this talk we present an implementation of combined modeling and solving in constraint programming (CP) and mixed integer programming (MIP) using the LP/MIP software XPRESS-MP and the finite domain CP solver SchedEns.
The approach has been applied successfully to several small and large-scale problems, a selection of which are described with some more detail.
The combined system implements a tight cooperation of the two techniques: a problem is represented in both software ("double modeling") and correspondences between variables of the two parts are established ("communication variables"). The combined search is defined on these communication variables. Based on the double model representation, each software constructs and directs its search tree in the way that is best suited for the corresponding representation.
For solving large-scale application problems where finding the optimal solution or proving optimality is prohibitively expensive in terms of running time, heuristics that complete the

current partial solution based on the Linear Programming relaxation lead in most cases to the best results.

Kurt Mehlhorn

Bound Consistency of the Sortedness Constraint

Bleuzen and Colmerauer gave an $O(n \log n)$ algorithm for narrowing the sortedness constraint. We give a new algorithm. The algorithm is simpler, can be made to run in time $O(N + n\alpha(n))$, where N is the largest integer appearing in the input, and links the problem to matching theory and the alldiff-constraint.
(joint work with Sven Thiel).

William R. Pulleyblank

Hilbert Bases and Cutting Planes

We review the theory of Hilbert bases, cutting planes and TDI-ness of polyhedra, including recent results of Eisenbrand and Bockmayr establishing the complexity of optimizing over the elementary closure of a polyhedron (in general, and for fixed dimension) and the counterexample due to Bruns, Gubeladze, Henk, Martin and Weismantel of the Caratheodory conjecture for Hilbert bases.
We apply these ideas to the cone generated by the incidence vectors of the node sets of the directed cycles of a digraph. We give a set of inequalities sufficient to define this cone, show that these incidence vectors form a Hilbert basis of the cone, and show that the integral Caratheodory property holds - every integral vector in the cone can be expressed as an integral linear combination of at most $|V|$ cycle incidence vectors.
This is joint work with Arlette Gaillard, Heinz Groeflin, and Alan J. Hoffman.

Laurent Michel

A Modeling Language for Local Search

Local search is a traditional technique to solve combinatorial search problems which has raised much interest in recent years. The design and implementation of local search algorithms is not an easy task in general and may require considerable experimentation and programming effort. However, contrary to global search, little support is available to assist the design and implementation of local search algorithms. This presents the design of *Localizer*, a modeling language which makes it possible to express local search algorithms in a notation close to their informal descriptions in scientific papers. Experimental results on several problems show the feasibility of the approach.

Robert E. Bixby

A Paradigm for Finding Cuts

Two approaches for generating cutting planes were discussed. The first was the classical Gomory mixed-integer cutting plane, introduced in the early 1960s. These cutting planes were long thought to be only of theoretical interest, but recent computational results demonstrate that they do indeed work very well in practice.
The second approach discussed was an apparently very general idea that was developed in the context of the traveling salesman problem in joint work with David Applegate, Vasek Chvatal, and Bill Cook. The idea is project the full problem down onto a problem of much smaller dimension, and then apply an optimization oracle together with general-purpose linear programming ideas to generate cutting planes for the original, full model.

Michael Trick

Integer Programming Models in Sports Scheduling

Sports Scheduling offers a number of interesting combinatorial problems that can be modeled either as integer or constraint programming models. Here we address two. The first is the minimum break problem where a schedule is given without the home/away pattern and the objective is to find the home/away pattern with the minimum number of breaks. Integer and constraint programming formulations build on each other to create improved solution techniques. The second problem is the Traveling Tournament Problem, a problem that seems quite difficult even for very small (six team) examples. Again, the interplay of constraint and integer programming leads to improved solution methods.