Structure, Duality, and Randomization: Common Themes in AI and OR

Carla P. Gomes
Computer Science Department
Cornell University
Ithaca, NY 14853
gomes@cs.cornell.edu

Abstract

Both the Artificial Intelligence (AI) community and the Operations Research (OR) community are interested in developing techniques for solving hard combinatorial problems. OR has relied heavily on mathematical programming formulations such as integer and linear programming, while AI has developed constrained-based search and inference methods. Recently, we have seen a convergence of ideas, drawing on the individual strengths of these paradigms. Furthermore, there is a great deal of overlap in research on local search and meta-heuristics by both communities. Problem structure, duality, and randomization are overarching themes in the study of AI/OR approaches. I will compare and contrast the different views from AI and OR on these topics, highlighting potential synergistic benefits.[1]

Introduction

In recent years we have seen an increasing dialog between the Artificial Intelligence (AI) and Operations Research (OR) communities, in particular in the area of combinatorial problems. These problems are ubiquitous and occur in areas as diverse as planning, scheduling, automated reasoning, and protein folding. AI approaches encompass a rich collection of knowledge representation formalisms for dealing with a wide variety of real-world problems. Some examples are constraint programming representations, logical formalisms, declarative and functional programming languages such as Prolog and Lisp, Bayesian models, rule-based formalism, etc. The downside of such rich representations is that in general they lead to intractable problems, and we therefore often cannot use such formalisms for handling realistic size problems. OR, on the other hand, has focused on more tractable representations, such as linear programming formulations. OR based techniques have demonstrated the ability to identify optimal and locally optimal solutions for well-defined problem spaces. In general, however, OR solutions are restricted to rigid models with limited expressive power. AI techniques, on the other hand, provide richer and more flexible representations of real-world problems, supporting efficient constraint-based reasoning mechanisms as well as mixed initiative frameworks, which allow the human expertise to be in the loop. The challenge lies in providing representations that are expressive enough to describe real-world problems and at the same time guaranteeing good and fast solutions.

Problem structure, *duality*, and *randomization* are overarching themes in the study of AI/OR approaches. In this paper I compare and contrast the different views from AI and OR on these topics, highlighting potential synergistic benefits.

Problem Structure

The ability to capture and exploit problem structure is of central importance, a way of taming computational complexity. In general structured models are easier to understand and compute with.

The OR community has identified several classes of problems with very interesting, *tractable*, structure. Linear Programming (LP) is a notable example of such a class. Linear Programming plays a major role in OR methods. Work done by Leonid Kantorovich in 1939 is considered the main precursor to Linear Programming (LP). In 1947, George Dantzig developed LP and the simplex method, initially conceived to speed up the process of providing a time-staged deployment, training and logistical program in military applications.[2] Interestingly, the word “programming” in Linear Programming has nothing to do with computer programming, but rather with the notion of “program” as used by the military to refer to plans of military operations. The simplex method made it possible to consider larger problems in areas as diverse as transportation, production, resource allocation, and scheduling problems.

The complexity of LP was not known for a long time. In the 70’s, Klee and Minty (1972) created an example that showed that the simplex method can require exponential time. However, despite its worst-case exponential complexity, the simplex method generally performs very well

[1]This paper is based on Gomes (2000).

[2]Ironically, Dantzig was not considered for the Nobel Prize in Economics for work related to the discovery and application of LP. The prize was given to Koopmans and Kantorovich for their work applying LP to problems in economics.

in practice. In the late 70's, Khachian (1979) developed a polynomial-time algorithm for linear programming. On practical problems, however, this method was much less efficient than the simplex method. In 1984, Karmarkar devised an interior point method that is more efficient and can outperform the simplex method on certain large problems instances. Still, the simplex method is often the method of choice. ILOG-CPLEX simplex based method, one of the leading LP commercial software, was shown to be very competitive or even outperforming interior point based methods on several benchmarks (Bixby 1999).

The main extensions of LP are Integer Programming (IP) and Mixed Integer Programming (MIP). IP and MIP extend LP to deal with integrality constraints and are the "bread and butter" of OR.

The standard OR approach for solving MIP problems is to use a branch-and-bound search. Branch-and-bound entails solving several LP's, which are relaxations of the original IP or MIP that provide guidance and tighten bounds for branch and bound techniques. First, an LP relaxation of the problem instance is considered. In such a relaxation, all variables of the problem are treated as continuous variables. If the solution to the LP relaxation problem has non-integer values for some of the integer variables, we have to branch on one of those variables. This way we create two new subproblems (nodes of the search tree), one with the floor of the fractional value and one with the ceiling. (For the case of binary (0/1) variables, we create an instance with the variable set to 0 and another with the variable set to 1.) Following the strategy of repeatedly fixing integer variables to integer values will lead at some point to a subproblem with an overall integer solution (provided we are dealing with a feasible problem instance). (Note we call any solution where all the integer variables have integer values an "integer solution".) In practice, it often happens that the solution of the LP relaxation of a subproblem already is an integer solution, in which case we do not have to branch further from that node. Once we have found an integer solution, its objective function value can be used to prune other nodes in the tree, whose relaxations have worse values. This is because the LP relaxation bounds the optimal solution of the problem. For example, for a minimization problem, the LP relaxation of a node provides a lower-bound on the best possible integer solution. Interestingly, branch-and-bound is a particular case of A*: the admissible heuristic is the LP relaxation.

Successful solutions of large-scale MIPs require formulations whose LP relaxations give a good approximation to feasible solutions. For instance, it is known that the Knapsack problem is relatively easy to solve if using the "right" LP formulations whose relaxations are very insightful for a branch and bound algorithm. However, some formulations of the Knapsack problem lead to poor relaxations of the corresponding LP, in the sense that they do not provide much information for a branch and bound algorithm.

Another very successful way of exploiting structure is the work of Dantzig and Wolfe on solving LP by means of *Decomposition*. It has had a major impact on solving large-scale problems. In fact, even though the simplex method can handle sparse problems with several thousands of rows quite comfortably, it does not scale up when it comes to truly huge problems. For such problems the simplex method is out of the question and the Dantzig-Wolfe decomposition is needed. An example of the application of such decomposition methods is column generation techniques. They have been successfully applied, *e.g.*, in Airline Crew Scheduling (see *e.g.*, Barnhart *et al.* 1994). Branch-and-price is an example of a column generation technique.

The crew scheduling problem is the problem of assigning crews to a given set of flights, in general all the flights of a specific fleet type. In this problem, sequences of flights (pairings) are assigned to crews so that each flight is assigned to exactly one crew. Since pairings are subject to complicated rules (safety and contractual rules) it would be difficult to express constraints and costs if a direct encoding were used.[3] Instead, valid pairings are enumerated and the problem is formulated as a set partioning problem (SPP). In this formulation each column or variable corresponds to a pairing and the objective is to partition all of the flights into a set of minimum cost pairings.

The main drawback is that the number of pairings grows exponentially with the number of flights. For example, Vance (1993) found more than 5 million valid pairings in a daily problem with 253 flights. Problems with 1000 flights, a typical size for a U.S. domestic carrier, are likely to have billions of pairings.

The approach used to solve this formidable problem uses Dantzig-Wolfe column generation. The LP relaxation of the SPP is solved, but only a subset of columns are initially considered. This problem is called the *restricted master problem*. New columns are generated only as needed, and if needed, and based on the information provided by the solution to the *restricted master problem, i.e.*, the *dual prices*. These *dual prices* allow one to determine which flights should be included in "good" columns for the master problem. The problem of generating new columns is called the *subproblem* or *pricing problem*.

Trans-Shipment Problems or *Network Flow Problems* are also notable examples of the importance of exploiting structure. Even though these problems are MIPs, which in general are NP-hard, the special structure of Network Flow Problems allows for very efficient (polynomial) algorithms. An interesting aspect of Network Flow Problems is that the optimal solution of instances involving only integral constraints are guaranteed to be also integer-valued. Many combinatorial problems, well beyond cases that deal with physical shipments of some commodity, such as scheduling problems, can be efficiently formulated as Network Flow Problems.[4]

Typically, when using OR methods, one starts by categorizing the problem into a class of problems for which efficient solution methods have been developed such as LP or

[3]By direct encoding we mean a formulation with variables x_{ij}, where $x_{ij} = 1$ if crew i is assigned to flight j.

[4]Unfortunately, Network Flow Algorithms cannot be used when there are global constraints on the nodes of the network. An example of a global constraint would state that the amount of goods shipped through certain nodes corresponds to 30 % of the total amount of goods shipped.

Network Flow. If the problem does not fit into such a class, one uses a more general formulation such as IP or MIP. At a second level, in general using an automated process, structure is detected using inference methods. For example, when solving IP's or MIP's, the derivation of "cutting planes" is very important to eliminate parts of the search space that are guaranteed not to contain the optimal solution. Cutting planes are linear inequalities that can be added to the original formulation of an IP with the guarantee that no integer solution will be eliminated, but with the advantage of eliminating fractional solutions generated by the linear relaxation of the problem. The addition of cutting planes leads to tighter relaxations, and therefore their solutions are better approximations of the IP's solution. Gomory (1958, 1963) pioneered this approach, showing how to systematically generate "cuts" that lead to an integer solution. "Cuts" or "cutting planes" are *redundant* constraints, in the sense that they do not eliminate feasible solutions. However, although these constraints are redundant in terms of the solution, they can play a major role during the search process. A classical example of the importance of cutting planes involves the pigeonhole problem: by adding the appropriate redundant constraints to a linear programming formulation, its relaxation immediately returns infeasibility. Without such redundant constraints, the results of the LP relaxation are useless. The OR community has developed several techniques for the generation of cuts, but, in general, it is not clear how to construct such cuts. A direction of research is the study of techniques that will lead to the generation of better cuts as well as efficient domain reduction techniques, and the combination of cuts with domain reduction techniques. Relevant work in this area is that of Lovasz and Schrijver (1991) and Balas, Ceria, and Cornuejols (1993). They have developed the *lift-and-project* technique. Hooker (1992) has developed cutting plane algorithms for IP and resolution methods in propositional logic. Work on the automated generation of cutting planes for problems such as the pigeonhole problem has been done by Barth (1996).

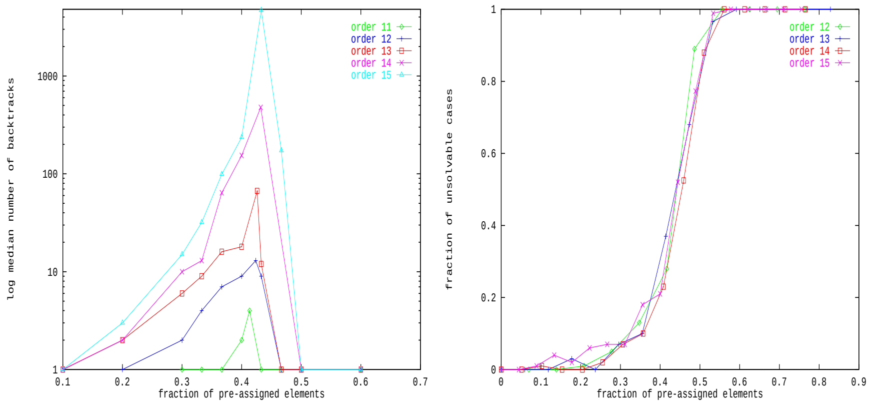
The OR community has identified several classes of problems with very a interesting, *tractable*, structure. LP, and Network Flow problems are good examples of such problems. OR also exploits the structure of problems during inference by generating "cutting planes", which allow for tighter relaxations that are therefore closer to the optimal integer solution. The AI community, on the other hand, has also identified interesting tractable problem classes: The most successful case, involving logical formalisms, is probably the Horn clausal form, which led to the development of expert systems.

The CSP community, on the other hand, has identified the special structure of several global constraints that are ubiquitous in several problems, which allow for the development of efficient constraint propagation techniques for the reduction of the variable domains. The CSP community mainly relies on domain reduction techniques for inference during search. A very successful strategy is to exploit the structure of special constraints and treat them as a global constraint (Beldiceanu and Contejean 1994, Caseau and Laburthe 1997; Regin 1994 and 1996). Some examples of such propagation methods are the constraint that guarantees that all elements of a vector are different (all-different constraint) and the constraint that enforces that certain values occur a given number of times in a given vector of variables (cardinality constraint). The implementation of such constraints is an interesting use of Network Flow algorithms (Regin 1994, 1996). The work done at Kestrel Institute using a transformational approach to scheduling encompasses the generation of very efficient constraint propagation techniques (Smith and Parra 1993). Dixon and Ginsberg (2000) combine satisfiability techniques from AI and OR, more specifically pseudo-Boolean representations and the cutting plane proof system with restricted learning methods such as relevance-bounded learning. They propose a new cutting plane proof for the pigeonhole principle of size n^2, and show how to implement intelligent backtracking techniques using pseudo-Boolean representation.

In general, however, the notion of structure is very hard to define, even though "we recognize structure when we see it." For example, there is not a methodology that shows how to construct good cutting planes. Therefore, formalizing the notion of structure and understanding its impact in terms of search is a key challenge for both AI and OR.

AI has made some progress in this area, namely in the study of phase transition phenomena, correlating structural features of problems with computational complexity. This is a new emerging area of research that is changing the way we characterize the computational complexity of NP-Hard problems, beyond the worst-case complexity notion: Using tools from statistical physics we are now able to provide a fine characterization of the spectrum of computational complexity of instances of NP-Complete problems, identifying typical *easy-hard-easy* patterns (Hogg et al. 1996). For example, in the Satisfiability problem it is known that the difficulty of problems depends on the ratio between number of clauses and number of variables (Kirkpatrick and Selman 1994).

We have studied phase transition phenomena in a more structured domain, the so-called Quasigroup Completion Problem (Gomes and Selman 1997). We introduced this domain to bridge the gap between purely random instances, such as Satisfiability, and highly structured problems, such as those from finite algebra (Fujita et al. 1993; Lam et al. 1989). The best way to view the quasigroup completion problem is in terms of the completion of a Latin square (which technically defines the multiplication table of the quasigroup). Given N colors, a Latin square is defined by an N by N table, where each entry has a color and where there are no repeated colors in any row or column. N is called the *order* of the square. The quasigroup completion problem (QCP) is the problem of whether a partially colored Latin square can be completed into a full Latin square by assigning colors to the open entries of the table. QCP is NP-complete (Colbourn 1984) and it has been used to study the effectiveness of a variety of local consistency measures for constraint satisfaction procedures (Stergiou and Walsh (1999a, 1999b), Walsh 1999, Regin 1994). (See www.cs.cornell.edu/gomes/ for a java applet demonstrating the quasigroup completion problem.)



(a) (b)

Figure 1: (a) Cost profile, and (b) phase transition for the quasigroup completion problem (up to order 15).

In our work we identified a phase transition phenomenon for QCP (Gomes and Selman 1997). At the phase transition, problem instances switch from being almost all solvable ("under-constrained") to being almost all unsolvable ("over-constrained"). The computationally hardest instances lie at the phase transition boundary. Figure 1 shows the median computational cost and phase transition. Along the horizontal axis we vary the ratio of pre-assigned colors. We note that even though all the instances are from an NP-complete problem, we clearly distinguish various regions of problem difficulty. In particular, both at low ratios and high ratios of preassigned colors the median solution cost is relatively small. However, in between these two regimes, the complexity peaks and, in fact, exhibits strong exponential growth. Somewhat surprisingly, the computational cost curves (curves for quasigroups of order 11, 12, 13, 14, 15) all peak at approximately at the same ratio, which is approximately 42% of the total number of cells of the matrix. [5]

The underlying phenomenon is explained by the left panel of figure 1 which shows the phase transition curve. The figure shows the probability of an instance being satisfiable as a function of the ratio of pre-assigned colors. At low ratios, almost all formulas are satisfiable; at high ratios, almost all formulas are unsatisfiable; whereas at the phase transition (around a ratio of 42%) 50% of the instances are satisfiable and 50% are unsatisfiable. This phase transition point directly corresponds to the peak in the computational complexity: This is a much finer characterization of the computational complexity of NP-complete problems than the standard worst-case exponential results (Cheeseman *et* al. 1991, Mitchell et al. 1992; Kirkpatrick and Selman 1994). (See Hogg et al. (1996) for a collection of papers on the topic.)

Another structural feature that has been recently formalized is the concept of *backbone*. The backbone of an instance corresponds to the shared structure of all the solutions of a problem instance. In other words, the set of variables and corresponding assignments that are common in all the solutions of a problem instance is the backbone (Monasson *et al.* 1999). We can link the computational hardness area to a phase transition which corresponds to a clear threshold phenomenon in the size of the backbone of the problem instances. The size of the backbone is measured in terms of the percentage of variables that have the same value in all possible solutions. We have observed a transition from a phase where the size of the backbone is almost 100% to a phase with a backbone of size close to 0%, for our quasigroup domain. The transition is sudden and it coincides with the hardest problem instances both for incomplete and complete search methods. Fig. 2 shows the backbone fraction as a function of the fraction of unassigned colors in satisfiable quasigroups (see also Achlioptas *et* al. 2000). The figure also includes the normalized cost of local search. The figure shows a sharp phase transition phenomenon in the backbone fraction, which coincide with the hardness peak in local search.[6]

The correlation between problem hardness and the appearance of the backbone suggests that search methods have the worst performance when a non-negligible fraction of the variables must appear in the backbone. When the backbone fraction nears 1, however, the problems are so constrained that incorrect choices near the root are quickly detected and corrected. For local search procedures, an explanation might be developed by considering the relationship between the backbone and set of solutions to the instances. When the backbone is small, there are many solutions widely distributed in the search space, and so local search may quickly

[5]The exact location of the phase transition appears to be characterized in terms of *Number-of-preassigned-colors*$/N^p$, where $p \neq 2$. However, and given that for low orders of quasigroups $p = 2$ is a good approximation, for simplification we talk about proportion of preassigned colors in terms of the total number of cells of the matrix, *i.e.,*N^2. See also Achlioptas *et al.* (2000).

[6]The figure gives data for $N = 36$. The hardness peak for our complete search method also lies in the phase transition region but is shifted slightly to the right. We are currently investigating whether that shift is real or part of the uncertainty in our data.

find one. When the backbone is near 1, the solutions are tightly clustered, so that that all clauses “vote” to push the search in the same direction. A partial backbone, however, may indicate that solutions are in different clusters that are widely distributed, with different clauses pushing the search in different directions. The role that backbone variables play in search is still not fully understood and requires further research. Nevertheless, we believe that the notion of backbone is critical when solving real-world problems. A good strategy for building real systems may encompass the use of powerful pre-processing techniques identifying backbone variables[7]. In practice, humans tend to implicitly use the notion of backbone variables (“critical resources”) when dealing, for example, with tasks such as scheduling and planning.

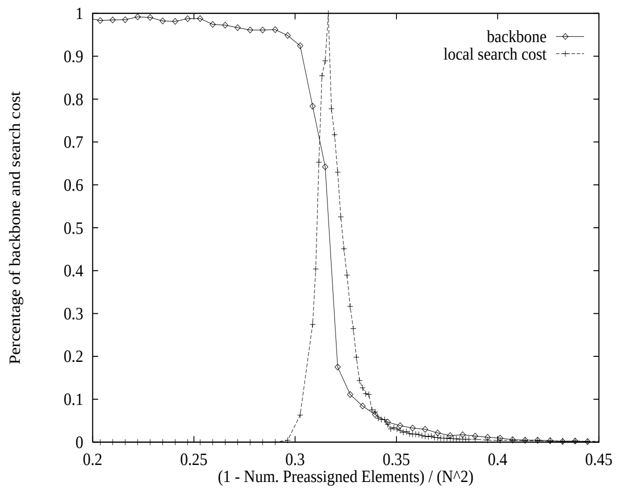


Figure 2: Backbone phase transition with cost profile.

Another interesting structural concept involves the identification of the tractable components of a problem. Monasson *et* al. (1999) introduced the 2+p SAT formulation to study the behavior of problems that are a mixture of 2-SAT and 3-SAT clauses. The proportion of 3-SAT clauses is defined by the parameter p. Note that this hybrid problem is NP-complete, for $p > 0$. However, and somewhat surprisingly, Monasson *et* al. showed that the problem scales linearly as long as the fraction of 3-SAT clauses is below 0.4. This is a promising result, suggesting that several problems that are NP-complete in the worst case may behave in a tractable way as long as a “good-size” well structured component is identified. Such a result makes it even more compelling for us to try to identify the well structured components of problems for which good efficient algorithms exist.

Duality

Duality plays an important role in OR. The basic idea is that, in general, problems can be considered both from a primal and dual perspective — maximizing the profit is equivalent to minimizing costs. Every maximization LP problem gives rise to a minimization LP problem, its *dual*. Interestingly, every feasible solution of one problem provides a bound on the optimal value of the other problem, and if one of them has an optimal solution, so does the other and their optimal values are the same. This is what the famous Duality Theorem states, formally proved by Gale *et al.* (1951). Its notions originated in conversations between Dantzig and von Neumann in the fall of 1947. The theory of duality is also used to perform sensitivity analysis and parametric analysis, *i.e.*, the study of the impact on the objective function when the level of resources (the right hand sides of the linear constraints) vary or when the coefficients of the objective function vary. The technique of *penalties* uses sensitivity analysis to tighten bounds during branch-and-bound search.

Duality is a powerful concept that has been extensively exploited by the OR community, a very elegant theory in the context of LP. There are dual theories for IP (Schrijver 1991), but, in general, as the problem grows they become very complex and are rarely used in practice. Furthermore, in general, duality is not yet well understood for problems involving constraints other than inequality constraints. Hooker *et al.* (2000) propose an interesting general approach to duality and sensitivity analysis that applies to both continuous and discrete problems. They generalize the classical idea of a dual to that of “inference dual”, which can be defined for any optimization problem. To solve the inference dual corresponds to obtaining a proof of the optimal value of the problem. Sensitivity analysis can be interpreted as an analysis of the role of each constraint in this proof. Traditional sensitivity analysis for LP is a special case of such an approach. Recently there have also been several promising results in the CSP community using dual formulation approaches (*e.g.*, to solve hard timetabling problems: McAloon *et al.* 1997 and Gomes *et al.* 1998b). Such approaches, by considering simultaneously two perspectives — the primal and dual view of the problem, allow for stronger inferences in terms of variable domain reductions. Research in this area, coupled with the study of the design of global constraints and good relaxation schemes for primal and dual formulations, is very promising. The study of new ways for performing sensitivity analysis based on duality is also a promising research area.

Randomization

Randomization and stochastic strategies have been very successful in local search methods. Local search methods or meta-heuristics are often used to solve challenging combinatorial problems. Such methods start with an initial solution, not necessarily feasible, and improve upon it by performing small “local” changes. One of the earliest applications of local search was to find good solutions for the Traveling Salesman Problem (TSP) (Lin (1965) and Lin and Kernighan (1973)). Lin and Kernighan showed that by performing successive swaps of cities to an arbitrary initial tour of cities, until no such swaps are possible, one can generate solutions that are surprisingly close to the shortest possible tour. There are several ways of implementing local search methods, depending on the choice of the initial so-

[7]Of course, the identification of all the backbone variables may be as difficult as solving the initial problem.

lution, types of “local” changes allowed, and feasibility and cost of (intermediate) solutions.

There is a great deal of overlap in research on local search by the AI and OR communities, namely in simulated annealing (Kirkpatrick *et al.* 1983), tabu search (Glover 1989), and genetic algorithms (Holland 1975). A recent new area of application for local search methods is in solving NP-complete *decision problems*, such as the Boolean satisfiability (SAT) problem. In 1992, Selman *et al.* showed that a greedy local search method, called GSAT, could solve instances with up to 700 hundred variables. Currently GSAT and variants (*e.g.*, WALKSAT) are among the best methods for SAT, enabling us to solve instances with up to 3000 variables (Selman *et al.* 1994). Closely related work in the area of scheduling is the technique of “MinConflicts” proposed by Minton *et al.* (1992).

Stochastic strategies have been very successful in the area of local search. However, local search procedures are inherently incomplete methods. An emerging area of research is the study of Las Vegas algorithms, *i.e.*, randomized algorithms that always return a model satisfying the constraints of the search problem or prove that no such model exists (Motwani and Raghavan 1995). The running time of a Las Vegas style algorithm can vary dramatically on the same problem instance. The extreme variance or “unpredictability” in the running time of complete search procedures can often be explained by the phenomenon of “heavy-tailed cost distributions” (Gomes *et al.* 2000). The understanding of these characteristics explains why “rapid restarts” and portfolio strategies are very effective. Such strategies eliminate the heavy-tailed behavior and exploit *any significant probability mass early on in the distribution.* Restarts and portfolio strategies in fact reduce the variance in runtime and the probability of failure of the search procedures, resulting in more robust overall search methods (Frost *et al.* 1997; Gomes and Selman 1999; Gomes *et al.* 1998a; Gomes *et al.* 2000; and Hoos 1999). So, somewhat counterintuitively, randomization actually provides a way for making solution methods more robust.

Conclusions

I have discussed the rich set of connections between artificial intelligence and operations research, focusing on approaches for dealing with hard combinatorial problems. We have seen how these connections can be grouped around three themes: *problem structure*, *duality*, and *randomization*. The overarching goal of these different themes in AI and OR is to uncover hidden tractable problem structure combined with a need for increased robustness and predictability of solution methods.

Acknowledgments

This work is funded by the Air Force Research Laboratory, Information Directorate and the Air Force Office of Scientific Research, under the New World Vistas Initiative (F30602-99-1-0005, F30602-99-1-0006, and AFOSR NWV project 2304, LIRL 97RL005N25).

References

Achlioptas, D., Gomes, C., Kautz, H. and Selman B. (2000) Generating satisfiable instances. *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, 2000.

Anderson, L. (1985). Completing partial Latin Squares. *Mathematisk Fysiske Meddelelser*, 41, 1985, 23–69.

Balas, E., Ceria, S., and Cornuejols, G. (1993) A lift and project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming* 58 (1993), 295-324.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1994) Branch-and-Price: column generation for solving huge Integer Programs. *Mathematical Programming. State of the Art* Birge, J., and Murty, K. (eds.), 1994, 186-207.

Barth,P. (1996) Logic based 0-1 Constraint Programming. Kluwer, 1996.

Beldiceanu N. and Contejean, E. (1994) Introducing global constraints in CHIP. *Mathl. Comput. Modelling*, 20 (12), 1994, 97–123.

Bixby R., (1999) MIP: Closing the gap between theory and practice. *19th IFIP TC7 Conference on System Modelling and Optimization* (Plenary Lecture), Cambridge, England, 1999.

Caseau, Y. and Laburthe, F. (1997) Solving various weighted matching problems with constraints. *Principles and Practice of Constraint Programming*, vol. 1330 of *Lecture Notes in Computer Science*, 1997, 17–31.

Caseau, Y., Laburthe, F., Le Pape, C., and Rottembourg B. (2000) Combining local and global search in a constraint programming environment. *Knowledge Engineering Review*, Vol. 15 (1), 2000.

Cheeseman, P., Kanefsky, B., and Taylor W. (1991) Where the really hard problems are. *Proc. IJCAI*, 1991.

Clements D., Crawford J., Joslin D., Nemhauser G., Puttlitz, M, and Savelsbergh, M. (1997) Heuristic optimization: a hybrid AI/OR approach. *Workshop of Constraint Programming*, Austria, 1997.

Colbourn, C. (1984). The complexity of completing Latin Squares. *Discrete Appl. Math.*, 8, (1984), 25-30.

Dantzig, G.B. (1991) Linear Programming. *History of Mathematical Programming, A collection of Reminiscences* Lenstra, J., Kan, R., Schrijver, A. (eds.), CWI, Amsterdam, 1991, 19-31.

Dantzig, G.B. and Wolfe, P. (1960) Decomposition principle for linear programs. *Operations Research*, 8, 1960, 101-111.

Denes, J. and Keedwell, A. (1974) Latin Squares and their applications. *Akademiai Kiado, Budapest, and English Universities Press*, London, 1974.

Dixon, H. and Ginsberg, M. (2000) Combining satisfiability techniques from AI and OR. *Knowledge Engineering Review*, Vol. 15 (1), 2000.

Frost, D., Rish, I., and Vila, L. (1997) Summarizing CSP hardness with continuous probability distributions. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 1997.

Fujita, M., Slaney, J., and Bennett, F. (1993). Automatic generation of some results in Finite Algebra. *Proc. IJCAI*, 1993.

Gale, D., Kuhn H, and Tucker A. (1951) Linear programming and the theory of games. *Activity Analysis of Production and Allocation,*, Wiley, New York, (1951) 317–329.

Glover, F. (1989) Tabu search — part I. *ORSA Journal on Computing*, 1(3) (1989) 190–206.

Gomes, C. (2000) Artificial Intelligence and Operations Research: challenges and opportunities in planning and scheduling. *Knowledge Engineering Review*, Vol. 15 (1), 2000.

Gomes, C.P., Kautz, H., and Selman, B. (1998a) Boosting combinatorial search through randomization. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

Gomes, C.P. and Selman, B. (1997) Problem structure in the presence of perturbations. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, New Providence, RI, 1997, 221–226.

Gomes, C.P. and Selman, B. (1999) Search strategies for hybrid search spaces. *Proceedings of the Eleventh International Conference on Tools with Artificial Intelligence (ICTAI-99)*, 1999.

Gomes, C.P., Selman, B., Crato, N, and Kautz, H. (2000) Heavy-Tailed phenomena in Satisfiability and Constraint Satisfaction Problems. *Journal of Automated Reasoning*, Vol. 24 (1/2) 2000, 67–100.

Gomes, C.P. and Selman, B., McAloon, K., and Tretkoff C. (1998b). Randomization in backtrack search: Exploiting Heavy-Tailed Profiles for Solving Hard Scheduling Problems. *Proc. AIPS-98*.

Gomory, R. (1958) An outline of algorithm for integer solutions to linear programs. *Bulletin of the American mathematical Society*, 1958, 64, 275–278.

Gomory, R. (1963) An algorithm for integer solutions to linear programs. *Recent Advances in Mathematical Programming* McGraw-Hill, Graves, R. and Wolfe, P. (eds.) 1963, 64, 260–302.

Holland, J.H (1992) *Adaptation in natural and artificial systems.* University of Michigan Pres, 1992.

Hooker, J. (1992) Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6, 1992, 271–286.

Hooker, J., Ottosson, G., Thorsteinsson, E. and Kim, H. (2000) A scheme for unifying optimization and constraint satisfaction methods. *Knowledge Engineering Review*, Vol. 15 (1), 2000.

Hoos, H. (1999) On the run-time behaviour of stochastic local search algorithms for SAT. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999, 661-666.

Jeroslow, R. (1980) A cutting plane game for facial disjunctive programs. *SIAM J. Control and Optimization*, 18, 1980, 264–280.

Kantorovich, V. (1939) Mathematical Methods in the organization of planning of production. *Management Science*, 6, 1960, 366–422 [English translation.]

Karmarkar, N. (1984) A new polynomial time algorithm for linear programming. *Combinatorica*, 4, 1984, 373–395.

Khachian, V. (1979) A polynomial time algorithm for linear programming. *Math. Doklady*, 20, 1979, 191–194. [English translation.]

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983) Optimization by simulated annealing. *Science*, 220 (1983) 671–680.

Kirkpatrick, S. and Selman, B. (1994) Critical behavior in the satisfiability of random boolean expressions. *Science*, 264 (May 1994) 1297–1301.

Klee, V. and Minty, G. (1972) How good is the simplex algorithm? *Inequalities-III* Shisha, O. (ed.) New York: Academic Press, 1972, 159–175.

Lam, C., Thiel, L., and Swiercz, S. (1989) The non-existence of finite projective planes of order 10. *Can. J. Math.*, Vol. XLI, 6, 1989, 1117–1123.

Lin, S. (1965) Computer solutions of the traveling salesman problem. *BSTJ*, 44, no 10 (1965), 2245–69.

Lin, S. and Kernighan, B.W. (1973) An effective heuristic for the traveling-salesman problem. *Oper. Res.* 21 (1973) 498–516.

Lovasz, L. and Schrijver A. (1991) Cones of matrices and set functions and 0-1 optimizations. *SIAM J. Control and Optimization*, 1991, 166–190.

Luby, M., Sinclair A., and Zuckerman, D. (1993). Optimal speedup of Las Vegas algorithms. *Information Process. Lett.*, 17, 1993, 173–180.

McAloon, K., Tretkoff C. and Wetzel G. (1997). Sports League Scheduling. *Proceedings of Third Ilog International Users Meeting*, 1997.

McAloon, K., Tretkoff C. and Wetzel G. (1998). Disjunctive programming and and cooperating solvers. *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*, 1998. Kluwer, Woodruff, D. (ed.), 75–96.

Mitchell, D., Selman, B., and Levesque, H.J. (1992) Hard and easy distributions of SAT problems. *Proc. AAAI-92*, San Jose, CA (1992) 459–465.

Minton, S., Johnston, M., Philips, A.B., and Laird, P. (1992) Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58 (1992) 161–205,

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., and Troyansky, L. (1996). Determining computational complexity from characteristic 'phase transitions'. *Nature*, Vol. 400(8), 1999.

Motwani, R. and Raghavan P. (1995) Randomized algorithms. Cambridge University Press, 1995.

Nemhauser, G., and Wolsey L. (1988) Integer and Combinatorial Optimization. John Wiley, New York, 1988.

Nemhauser, G., and Trick, M. (1997) Scheduling a major college basketball conference. Georgia Tech., Technical Report, 1997.

Oddi A. and Smith, S. (1997) Stochastic procedures for generating feasible schedules. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, New Providence, RI, 1997.

Puget, J-F., and Leconte, M. (1995). Beyond the black box: constraints as objects. *Proceedings of ILPS'95*, MIT Press, 513–527.

Regin J.C. (1994). A filtering algorithm for constraints of difference in CSPs. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-94)*, Seattle, 1994.

Regin J.C. (1996). Generalized arc consistency for global cardinality constraint. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Oregon, 1996.

Rodosek, R., and Wallace, Mark (1998). One model and different solvers for hoist scheduling problems. Manuscript in preparation.

Schrijver A. (1991) Theory of linear and integer programming. Wiley, 1986.

Selman, B., Kautz, H., and Cohen, B. (1994) Noise strategies for improving local search. *Proc. AAAI-94*, Seattle, WA (1994) 337–343.

Selman, B., Levesque, H.J., and Mitchell, D. (1992) A new method for solving hard satisfiability problems. *Proc. AAAI-92*, San Jose, CA (1992) 440–446.

Smith D., and Parra E. (1993). Transformational approach to transportation scheduling. *Proceedings of the Eigth Knowledge-Based Software Engineering Conference*, 1993, Chicago,

Stergiou, K. and Walsh, T. (1999a) The Difference All-Difference Makes Proc. of *IJCAI-99*, Stockholm, Sweden.

Stergiou, K. and Walsh, T. (1999b) Encodings of non-binary constraint satisfaction problems. Proc. *AAAI-99*, Orlando, FL. 1999.

Vance, P., (1993) Crew scheduling, cutting stock, and column Generation: solving huge integer programs. Georgia Tech., PhD Thesis, 1993.

Walsh, T. (1999) Search in a Small World. Proc. of *IJCAI-99*, Stockholm, Sweden, 1999.