In Proc. AIPS98, Pittsburg, PA, June 1998.

Randomization in Backtrack Search: Exploiting Heavy-Tailed Profiles for Solving Hard Scheduling Problems

Carla P. Gomes* **Bart Selman**
Computer Science Department
Cornell University
Ithaca, NY 14853
{gomes,selman}@cs.cornell.edu

Ken McAloon **Carol Tretkoff**
Ilog, Inc.
Mountain View, CA 94043
{mcaloon,tretkoff}@ilog.com

Abstract

We study the runtime profiles of complete backtrack-style search methods applied to hard scheduling problems. Such search methods often exhibit a large variability in performance due to the non-standard nature of their underlying cost distributions. The distributions generally exhibit very long tails or "heavy tails" and are best characterized by a general class of distributions that have no moments (*i.e.*, an infinite mean, variance, etc.). We show how one can exploit the special nature of such distributions to significantly improve upon deterministic complete search procedures.

Introduction

Combinatorial search methods often exhibit a remarkable variability in performance. For example, we see significant differences on runs of different heuristics, runs on different problem instances, and, for stochastic methods, runs with different random initial seeds. The inherent exponential nature of the search process appears to magnify the unpredictability of search procedures. It is not uncommon to observe a combinatorial method "hang" on a given instance, whereas a different heuristic, or even just another stochastic run, solves the instance quickly.

In this paper, we will show how one can take advantage of this extreme variability of complete combinatorial search methods. In particular, we will show how one can improve the performance of a deterministic complete backtrack-style search method by introducing a stochastic element, while maintaining completeness. We demonstrate the effectiveness of our strategy on a class of known hard scheduling problems, derived from real-world timetabling problems. These problems have been studied extensively in the Operations Research community using integer programming (IP) methods.

First we introduce a Constraint-Programming (CSP) formulation of our problem domain. This formulation scales better than the best IP formulations. We then show how one can yet further improve upon our CSP approach by adding a stochastic element to the deterministic search strategy. We should stress that our method remains complete, unlike, for example, stochastic local search strategies. The effectiveness of our approach can be explained in terms of the heavy-tailed nature of the underlying cost distributions of complete backtrack-style search procedures. That is, the distributions are characterized by extreme outliers relative to the median cost value. This phenomenon manifests itself in terms of the long-tails of the cost distributions and the highly erratic behavior of the mean search cost over multiple runs. Given its simplicity and generality, our approach can be easily adapted to improve the performance of other backtrack-style search methods used in planning and scheduling.

Problem Domain

We consider problems derived from sports scheduling applications. The literature in this area is growing, and one can begin to get a sense of the range and mathematical difficulty of the problems encountered (de Werra 1988; Nemhauser and Trick 1997; and Schreuder 1980; 1992). In sports scheduling problems one of the issues is timetabling, where by timetabling we mean determining the existence of a feasible schedule that takes into consideration constraints on how the competing teams can be paired, as well as how each team's games are distributed in the entire schedule. In particular, we consider the timetabling problem for a "round robin" schedule: every team must play every other team exactly once. The *global* nature of the pairing constraints makes this a particularly hard combinatorial search problem.

Typically, a game will be scheduled on a certain field or court, at a certain time, etc. This kind of combination will be called a slot. These slots can vary in desirability due to such factors as lateness in the day, the location and the condition of the field, etc. The problem is to schedule the games such that the different periods are assigned to the teams in an equitable manner over

*Carla P. Gomes is also a research associate of the Air Force Research Laboratory, Rome, NY, USA.

the course of the season. The round robin scheduling problem considered here is something of a "classic" in the operations research community, because it presents a very tough challenge for integer programming methods.

The problem is formally defined as follows:

1. There are N teams (N even) and every two teams play each other exactly once.
2. The season lasts $N-1$ weeks.
3. Every team plays one game in each week of the season.
4. There are $N/2$ periods and, each week, every period is scheduled for one game.
5. No team plays more than twice in the same period over the course of the season.

The meeting between two teams is called a *matchup* and takes place in a *slot*, *i.e.*, in a particular period in a particular week. For example, a valid schedule for 8 teams named 0,1,2,3,4,5,6,7 would be given by filling in the slots with matchups as in Table 1.

The 8 team problem instance is relatively simple and can be done by brute force if necessary. However, the combinatorics of this scheduling problem are explosive. For an N team league, there are $N/2 \cdot (N-1)$ matchups (i, j) with $0 \leq i < j < N$ to be played. A schedule can be thought of as a permutation of these matchups. So, for N teams the search space size is $(N/2 \cdot (N-1))!$, *i.e.*, the search space size grows as the factorial of the square of $N/2$. This means that algorithms cannot be expected to scale nicely; and, as we shall see, they do not.

IP And CSP Formulations

When formulating the timetabling problem, one can adopt two complementary ways of representing the problem. We will refer to these two representational approaches as the primal model and the dual model. In the primal model, the goal is to fill in the different periods of the schedule with matchups. The dual model takes the complementary perspective, *i.e.,* it starts with the matchups, and looks for the periods to place them in.

IP Formulation

Integer Programming (IP) is an important tool for the timetabling phase of sports scheduling problems. The 0-1 integer programming model for the primal approach can be summarized as follows. For each pair of teams (i,j) with $0 \leq i < j < N$, for each row k and each column m, there is a binary variable $x_{i,j,k,m}$ which will be 1 if i plays j in the slot in row k and column m and 0 otherwise. (We're considering a row/column representation as used in Table 1.) The constraints are:

- for all i and j: $\sum_{k,m} x_{i,j,k,m} = 1$
  (each team plays each other team exactly once)
- for all i and m: $\sum_{j,k} (x_{i,j,k,m} + x_{j,i,k,m}) = 1$
  (team i plays once in column m)
- for all i and k: $\sum_{j,m} (x_{i,j,k,m} + x_{j,i,k,m}) \leq 2$
  (each team plays at most twice in a period)
- for all k and m: $\sum_{i,j} x_{i,j,k,m} = 1$
  (each slot has one game)

This 0-1 integer program is elegant in its simplicity. However, it suffers from the fact that as N increases the number of 0-1 variables and the number of constraints grows quite rapidly: we have $O(N^4)$ 0-1 variables and $O(N^2)$ linear constraints each of size $O(N^2)$.

CSP Formulation

Our Constraint Satisfaction Formulation (CSP) of the sports scheduling problem starts with the primal model. Here, we assign a pair of teams to each valid slot of the sports season. Given N teams, the season lasts $(N-1)$ weeks and there are $N/2$ periods per week. Our representation uses $2*(N-1)*N/2$ variables that have to be assigned a team (a value from $\{0, 1, 2, \ldots, N-1\}$). The number of variables corresponds to twice the number of slots in the season, since two teams play in each slot.

To solve the CSP problem we use a complete backtrack-style algorithm with the first-fail heuristic for variable assignment. In the first-fail heuristic one selects the variables with the smallest domain first. It's one of the most effective, general heuristics. We encoded this problem in C++ using ILOG SOLVER, a C++ constraint programming library (Puget and Leconte 1995). ILOG enables us to express constraints declaratively, and provides a backtracking mechanism that supports constraint propagation.

In order to facilitate the representation of the constraints, we use several auxiliary variables, linked to the main variables. For example, we represent the schedule both as row vectors and as column vectors, mimicking the table format. Each row corresponds to $2*(N-1)$ variables, since it corresponds to the number of weeks required for the whole season, $(N-1)$ times two, accounting for two teams per slot. With this explicit representation of the row, it is straightforward, for example, to associate with each row the constraints that a team cannot play the same period more than twice during the whole season. Similarly, with the column view of the schedule we can easily represent the constraint that the same team can only play one match a week. In order to enforce that all the matchups have to be distinct, we define a function that associates a single number with each pair of teams, and we introduce a constraint that asserts that each pair number can only occur once in the entire schedule.

In order to increase propagation, we also represent explicitly the dual view of the schedule, namely the assignment of different slots to the different matchups:

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3

Table 1: An 8 Team Round Robin Timetable

whenever an assignment and propagation is performed in one of the representations, the corresponding propagation is also triggered for the other representation. We use an antisymmetry strategy that fixes the matchups in some of the slots in order to reduce the number of solutions that are the result of simple permutations of an answer.

As we will discuss in the next section, the results obtained with our CSP formulation are superior to the results that we obtained with the IP formulation. This superiority is due to the use of CSP methods for this type of problem. First of all, as has already been pointed out, the number of variables that are needed is substantially smaller than in an IP. The strength of the CSP formulation is that it can deal directly with constraints that "punch holes" in the domains of variables; this makes for highly "non-convex" domains. Our CSP formulation also has the advantage of not being limited to 0-1 variables, and it allows us to easily model both the primal and dual approaches. The combination of both representations proved very efficient since it results in increased level of propagation of constraints with fast pruning of the search space.

Experiments and Randomization

Using the IP formulation, we can find a solution for $N = 12$ in about 14 hours; we were unable to find a solution for $N = 14$.[1] Our CSP formulation dramatically improves upon these numbers. It gives us a solution for $N = 12$ in about 13 seconds and for $N = 14$ in about 411 seconds.[2] We could not find a solution for $N = 16$. (It ran for over 48 hours.)

As is clear from these benchmarks, the problem quickly becomes very difficult, even for moderate values of N. Apparently, the subtle interaction between global and local constraints makes the search for a globally consistent solution surprisingly hard.

Our next goal is to further enhance the performance of our CSP based approach. To do so, we consider adding a stochastic element to our backtrack-style search. In previous work, in the context of a highly structured search domain based on algebraic structures (Gomes *et al.* 1997), we found that randomized versions of complete search methods exhibit a surprising variability between runs on the same instance with different random initial seeds. In fact, the number of backtracks required to find a solution can vary over many orders of magnitude. The key issue is whether we can effectively take advantage of such large variations between runs to solve previously unsolved instances of the timetabling problem.

To introduce a stochastic element in our search procedure, we break ties randomly in our first-fail heuristic (as opposed to the standard, deterministic "lexicographically-first" approach). It is not difficult to devise more elaborate schemes, but — somewhat to our surprise — we found this simple randomization scheme to be quite effective. We also introduce a parameter (the "cutoff" parameter), which specifies after how many backtracks we restart our backtrack search from the root of the tree with a new random seed. This simple modification of the deterministic strategy leads to a rather dramatic change in overall behavior. Note that we maintain completeness since we gradually increase the cutoff. In the limit, as the cutoff tends to infinity, we reach the original complete randomized backtracking procedure. This argument is akin to that used for iterative deepening search.

We ran our randomized backtrack search method with a cutoff of 10^5 on our scheduling problem with $N = 16$. In 100 runs, our procedure found a solution in 6 of them. The average number of backtracks on the successful runs was 34,546, with the shortest run taking only 914 backtracks, and the longest run 48,534. Taking into account the time spent on unsuccessful runs, the randomized procedure finds a solution in 2 hours, on average. (Note again that our deterministic procedure did not find a solution in 48 hours.) For the $N = 18$ case, we ran with a cutoff of 5.10^5, and found a solution after approximately 22 hours.

These results show that introducing a stochastic element in a backtrack-style search procedure can directly enhance its performance. In fact, as we see here, it allows us to solve previously unsolved problem instances. It should be noted that recently there has been a lot of interest in this sports timetabling problem (McAloon *et al.* 1997). Since the submission of this paper, a lot of progress has been made in terms of solving larger instances (McAloon *et al.* 1998). By using multiple threads on a 14 processor Sun system, 26 and 28 teams schedules were generated, which is the record as of this writing (Wetzel and Zabatta, in preparation). We believe these numbers can be improved upon with our

[1] Experiments ran on a Sun UltraSparc using CPLEX, a leading commercial integer programming package.

[2] Experiments ran on a 200MHz SGI Challenge using ILOG SOLVER, a prominent constraint programming package. The speed of the SGI is comparable to that of a Sun UltraSparc.

randomization technique.

In the next section, we will explain the success of our approach in terms of the special nature of the underlying cost distribution of backtrack-style search procedures. We believe this is the first such explanation, even though, the "power of randomization" in combinatorial search has been informally recognized by others (for recent work in scheduling domains, see e.g., Bresina 1996 and Oddi and Smith 1997).

Heavy-Tailed Distributions

In order to obtain a better understanding of the behavior of our randomized strategy, we now consider some detailed statistics as gathered on smaller problems. Figure 1 shows the cumulative cost distribution (F(x)) for $N = 12$. The cost is given in terms of number of backtracks to find the first solution. The figure gives aggregate data over 10,000 runs. A striking feature of the cost distribution is its surprisingly long tail: even though the median cost is less than 2000 backtracks, about 5% of the runs take over 20,000 backtracks, and about 1% of the runs are not solved after 200,000 backtracks. Clearly, the long tail suggests a form of "thrashing" behavior that should be avoided. We can use the "cutoff" parameter to do so. The data also shows that the shortest 5% of the runs only take about 200 backtracks.

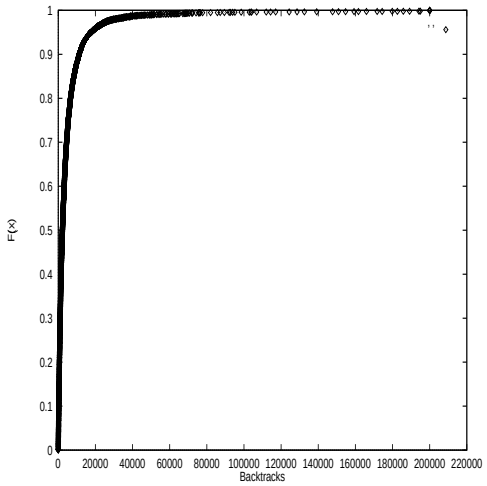


Figure 1: Heavy-tailed behavior.

In order to model the long tail behavior of our distribution, we consider distributions which asymptotically have tails of the Pareto-Lévy form, *viz.*

$$\Pr\{X > x\} \sim C.x^{-\alpha}, \quad x > 0 \tag{1}$$

where $\alpha > 0$ is a constant (Mandelbrot 1960; and Samorodnitsky 1994). These are "heavy-tailed" distributions, *i.e.*, distributions whose tails have a *power law decay*. The constant α is called the *index of stability* of the distribution. For $\alpha < 2$, moments of X of order less than α are finite while all higher order moments are infinite, *i.e.*, $\alpha = \sup\{a > 0 : \mathrm{E}|X|^a < \infty\}$. For example, when $\alpha = 1.5$, the distribution has a finite mean but no finite variance. With $\alpha = 0.6$, the distribution has neither a finite mean nor a finite variance.

In order to check for the existence of heavy tails in our distributions, we proceed in two steps. First, we graphically analyze the tail behavior of the sample distributions. Second, we formally estimate the index of stability.

If a Pareto-Lévy tail is observed, then the rate of decrease of the estimated density is a power law. (Standard distributions exhibit exponential decay.) From (1), we have $1 - F(x) = \Pr\{X > x\} \sim C.x^{-\alpha}$, so the complement-to-one of the cumulative distribution, $F(x)$, also decays according to a power law.

Given the power law decay of the complement-to-one of the cumulative distribution of a heavy-tailed random variable, its log-log plot should show an approximately linear decrease in the tail. Moreover, the slope of the observed linear decrease provides an estimate of the index α. In contrast, for a distribution with an exponentially decreasing tail, the log-log plot should show a faster-than-linear decrease in the tail.

Figure 2 shows the log-log plot of the complement-to-one of the cumulative distribution, 1-F(x), for our timetabling problem ($N = 12$). Plot 2a gives the full range of distribution, while plot 2b shows the tail of the distribution ($X > 10,000$). The linear nature of the tail in plot 2b directly reveals tails of the Pareto-Lévy type.

For contrast, in Figure 3, we show the log-log plot of a gamma and a normal distributed variable. It is clear from the plots that these distributions do not exhibit heavy-tailed behavior, given the faster-than-linear decrease in the tails.

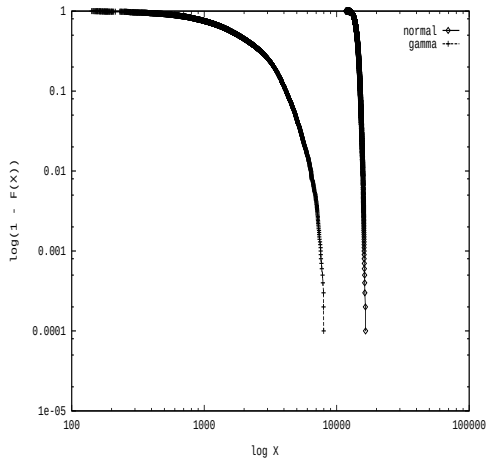
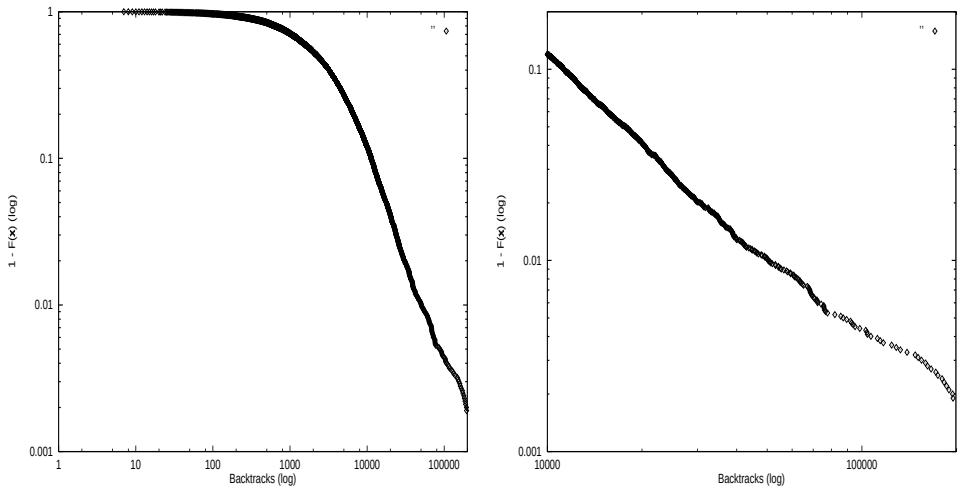


Figure 3: No heavy-tails.

To complement our visual check of heavy-tailed behavior of Figure 2, we calculate the maximum likelihood estimate of the index of stability (the value of α): For our timetabling problem, for $N = 12$, we obtain $\alpha = 0.7$, which is consistent with the hypothesis of infinite mean and infinite variance, since $\alpha \leq 1$.



(a) (b)

Figure 2: Log-log plot of heavy-tailed behavior.

cutoff	succ. rate	exp. cost ($\times 10^6$)
200	0.0001	2.2
5,000	0.003	1.5
10,000	0.009	1.1
50,000	0.07	0.7
100,000	0.06	1.6
250,000	0.21	1.2
1,000,000	0.39	2.5

Table 2: Solving $N = 16$ for a range of cutoff values.

For problems for which we can empirically determine the overall cost profile, we can calculate an *optimal* cutoff value to minimize the expected cost of finding a solution. However, our interest is in solving previously unsolved instances, such as the $N = 16$ and $N = 18$ case. These problems are too hard to obtain a cost distribution. For example, for $N = 16$, running with a cutoff of 1,000,000 gives a success rate of less than 40%, so we do not even reach the median point of the distribution. Each run takes about 2 hours to complete. (We estimate that the median value is around 2,000,000. Our deterministic procedure apparently results in a run that still lies to the right of the expected median cost.) In order to find a good cutoff value for very hard problem instances, the best available strategy is a trial-and-error process, where one experiments with various cutoff values, starting at relatively low values. The optimal cutoff for these problems does lie below the "median cutoff". This can be seen from Table 2, which gives the expected cost (backtracks) for finding a solution for $N = 16$ for a range of cutoff values. The optimal cutoff is around 5.10^4, resulting in an expected cost per solution of 7.10^5 backtracks.

So far, we have identified heavy-tailed behavior of the cost distribution to the right of the median. The heavy tail nature shows that there is a computationally significant fraction of very long runs, decaying only at a polynomial rate. The strategy of running the search procedure with a cutoff less than the median value of the distribution clearly avoids these long runs in the tail.

Our experiments also suggest a heavy tail phenomenon on the left-hand side of the median value of the cost distribution, which means that the success rate for a solution only increases polynomially with the number of backtracks. This explains how a relatively low cutoff value still gives a sufficiently high success rate to allow us to solve the problem instance. For example, for $N = 16$, we observed several runs that took less than 200 backtracks, compared to a median value of around 2,000,000. For $N = 18$, we ran with a cutoff of $500,000$ and solved the instance after 20 tries. Each try took about 1 hour, and the successful run took 350,632 backtracks. Figure 4 displays the log-log plot of the left-hand side of the cumulative distribution for $N = 14$. Its linear nature is an indication of heavy-tailed behavior of the left-hand side of the distribution.

In general, we conjecture that α for the tail on the left is less than 1.0 on hard combinatorial search problems. This conjecture has strong implications in terms of algorithm design: It means that in order to obtain the minimal expected run time, a preferred strategy consists of relatively short runs of a randomized backtrack-style procedure. More extensive experiments on other domains are needed to confirm this conjecture.

Conclusion

We have shown that one can exploit the heavy-tailed nature of a randomized backtrack-style algorithm to solve hard timetabling problems. The heavy-tailed nature of the underlying cost distribution implies a relatively high frequency of "outliers" on *both* sides of the median of the distribution. These outliers suggest a

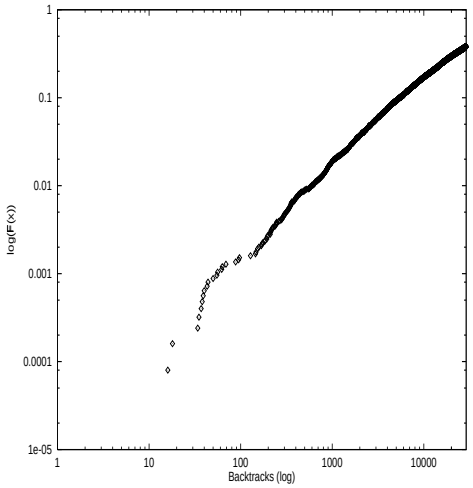


Figure 4: Log-log plot of heavy-tailed behavior on the left-hand side.

strategy with a low cutoff value: this avoids extremely long runs on the right-hand side of the distribution, and, moreover, exploits the occurrence of very short runs on the left. Using such a randomization strategy, we were able to solve hard round-robin timetabling instances of up to size 18, when the corresponding deterministic version could only handle instances up to size 14. We believe that the generality of our approach will lead to further applications in other planning and scheduling domains.

Acknowledgments

We would like to thank the reviewers for their comments. The first author is funded by the Air Force Research Laboratory, Rome Laboratory and the Air Force Office of Scientific Research, under the New World Vistas Initiative (F30602-97-C-0037 and AFOSR NWV project 2304, LIRL 97RL005N25).

References

Alt, H., Guibas, L., Mehlhorn, K., Karp, R., and Wigderson A. (1996) A method for obtaining randomized algorithms with small tail probabilities. *Algorithmica*, 16, 1996, 543–547.

Bresina, J. (1996) Heuristic-biased stochastic sampling. *Proc. AAAI-96*, Portland, OR, 1996.

Chambers, John M., Mallows, C.L., and Stuck, B.W. (1976) A method for simulating stable random variables. *Journal of the American Statistical Association* 71, 340–344.

Crovella, M., Taqqu, M.S., and Bestavros, A. (1997) Heavy-tailed probability distributions in the World Wide Web. In *A practical guide to heavy tails: Statistical techniques for analyzing heavy tailed distributions*, R. Adler, R. Feldman, and M.S. Taqqu (eds.), Birkhauser, Boston (1997).

deWerra, D. (1988) Some models of graphs for scheduling sports competitions, *Discrete Applied Mathematics* 21 (1988), 47-65.

Frost, D., Rish, I., and Vila, L. (1997) Summarizing CSP hardness with continuous probability distributions. *Proc. AAAI-97*, New Providence, RI, 1997, 327–333.

Gomes, C.P. and Selman, B. (1997) Problem structure in the presence of perturbations. *Proc. AAAI-97*, New Providence, RI, 1997, 221–226.

Gomes, C.P., Selman, B., Crato, N. (1997) Heavy-Tailed Distributions for Combinatorial Search. *Proc. Constraint Programming*, Linz, Austria, Nov. 1997.

Hall, P. (1982) On some simple estimates of an exponent of regular variation. *Journal of the Royal Statistical Society,* B 44, 37–42.

Hill, B. (1975) A simple general approach to inference about the tail of a distribution. *Annals of Statistics*, 3, 1975, 1163–1174.

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

McAloon, K., Regin, J-C., Tretkoff C. and Wetzel G. (1998). Constraint-Based Programming for Sports League Scheduling. Manuscript in preparation 1998.

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

Mandelbrot, Benoit B. (1960) The Pareto-Lévy law and the distribution of income. *International Economic Review* 1, 79–106.

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. *Proc. 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.

Samorodnitsky, Gennady and Taqqu, Murad S. (1994) *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*, Chapman and Hall, New York.

Schreuder, J.A.M. (1980) Constructing Timetables for Sport Competitions, *Mathematical Programming Study* 13 (1980), 58-67.

Schreuder, J. A. M. (1992) Combinatorial Aspects of Construction of Competition Dutch Professional Football Leagues, *Discrete Applied Mathematics* 35 (1992) 301-312.

van Hentenryck, P. , Deville, Y., and Teng Choh-Man (1992) A generic arc consistency algorithm and its specializations. *Artificial Intelligence*, 57, 1992.

Veloso, M. (1992). Learning by analogical reasoning in general problem solving. Ph.D. Thesis, CMU, CS Techn. Report CMU-CS-92-174.

Zolotarev, V.M. (1986) One-dimensional Stable Distributions, Vol. 65 of "Translations of mathematical monographs", American Mathematical Society. Translation from the original 1983 Russian Ed.