Annals of Operations Research 130, 117–141, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Approximations and Randomization to Boost CSP Techniques

CARLA P. GOMES gomes@cs.cornell.edu
Faculty of Computing and Information Science and Department of Applied Economics and Management, Cornell University, Ithaca, NY 14853, USA

DAVID B. SHMOYS shmoys@cs.cornell.edu
Department of Computer Science and School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA

Abstract. In recent years we have seen an increasing interest in combining constraint satisfaction problem (CSP) formulations and linear programming (LP) based techniques for solving hard computational problems. While considerable progress has been made in the integration of these techniques for solving problems that exhibit a mixture of linear and combinatorial constraints, it has been surprisingly difficult to successfully integrate LP-based and CSP-based methods in a *purely* combinatorial setting. Our approach draws on recent results on approximation algorithms based on LP relaxations and randomized rounding techniques, with theoretical guarantees, as well on results that provide evidence that the runtime distributions of combinatorial search methods are often heavy-tailed. We propose a complete randomized backtrack search method for combinatorial problems that tightly couples CSP propagation techniques with randomized LP-based approximations. We present experimental results that show that our hybrid CSP/LP backtrack search method outperforms the pure CSP and pure LP strategies on instances of a hard combinatorial problem.

1. Introduction

In recent years we have seen the development of successful methods for solving optimization problems by integrating techniques from Constraint Programming (CP) and Operations Research (OR) (see, e.g., (Gomes, 2001)). Such hybrid approaches draw on the individual strengths of these different paradigms: OR heavily relies on mathematical programming formulations such as integer and linear programming, while CP uses constrained-based search and inference methods. This is particularly true in domains where we have a combination of linear constraints, well-suited for linear programming (LP) formulations, and discrete constraints, suited for constraint satisfaction problem (CSP) formulations (see, e.g., (Little and Darby-Dowman, 1995; Bockmayr and Kasper, 1998; Heipcke, 1998; Guéret and Jussien, 1999; Hooker et al., 1999; Rodosek, Wallace, and Hajian, 1999; Caseau and Laburthe, 1995; Focacci, Lodi, and Milano, 1999a, 1999b, 2000; Refalo, 1999, 2000)). Nevertheless, in a *purely combinatorial* setting, with only discrete, non-numerical variables, so far it has been surprisingly difficult to integrate LP-based and CSP-based techniques. For example, despite a significant amount of work on using LP relaxations to solve Boolean satisfiability (SAT) problems, practical state-of-the-art solvers do not incorporate LP relaxation techniques. From a practical point of view, the challenge is how to integrate such techniques into practical solvers. The basic idea is to use the information from LP relaxations to guide the combinatorial search process. A key issue is whether the LP relaxation provides sufficient useful additional information – in particular, information that is not easily uncovered by constraint propagation and inference techniques (see, e.g., (Hooker, 1996; Kamath et al., 1992; Warners, 1999)). Of course, also the cost of solving the LP relaxation should not outweigh the benefits in the reduction of search cost.

Another aspect to note is that while LP relaxations combined with several heuristic strategies have been used extensively by both the OR and CP communities, in general, such heuristic approaches do not provide rigorous guarantees concerning the quality of the solutions that they achieve. A more theoretically well-founded approach can be obtained by using approximation algorithms with formal guarantees on solution quality and efficiency.

The area of approximation algorithms is a new and very active research area. Approximation algorithms are procedures that provide a feasible solution in polynomial time. Note that in most cases it is not difficult to devise a procedure that finds some solution. However, we are interested in having some guarantee on the quality of the solution, a key aspect that characterizes approximation algorithms. The quality of an approximation algorithm is the "distance" between its solutions and the optimal solutions, evaluated over all the possible instances of the problem. Informally, an algorithm approximately solves an optimization problem if it always returns a feasible solution whose measure is close to optimal, for example within a factor bounded by a constant or by a slowly growing function of the input size. More formally, given a maximization problem Π and a constant α $(0 < \alpha < 1)$, an algorithm $\mathcal{A}$ is an α-approximation algorithm for Π if its solution is at least α times the optimum, considering all the possibles instances of problem Π. Interestingly, standard algorithm design techniques such as greedy and local search methods, dynamic programming, and classical methods of discrete optimization such as linear programming and semidefinite programming are key tools to devise good approximation algorithms. Randomization is also a powerful tool for designing approximation algorithms. So, while approximation algorithms are, in general, based on standard algorithm design techniques, the performance guarantee associated with them is novel, which is not provided by heuristic methods.

Our approach for solving purely combinatorial problems draws on recent results on some of the best approximation algorithms with theoretical guarantees based on LP relaxations and randomized rounding techniques (see, e.g., (Chudak and Shmoys, 1999; Motwani, Naor, and Raghavan, 1997)), as well on results that uncovered the extreme variance or "unpredictability" in the runtime of complete search procedures, often explained by so-called heavy-tailed cost distributions (Gomes, Selman, and Crato, 1997; Gomes et al., 2000; Walsh, 1999). More specifically, we propose a *complete* randomized backtrack search method that tightly couples CSP propagation techniques with randomized LP-based approximations.

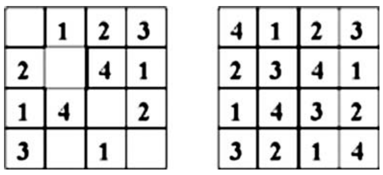
We use as a benchmark domain a *purely* combinatorial problem, the quasigroup (or Latin square) completion problem (QCP). Each instance consists of an n by n matrix with n^2 cells. A complete quasigroup consists of a coloring of each cell with one of n colors in such a way that there is no repeated color in any row or column. Given a partial coloring of the n by n cells, determining whether there is a valid completion into a full quasigroup is an NP-complete problem (Colbourn, 1984). The underlying structure of this benchmark is similar to that found in a series of real-world applications, such as timetabling, experimental design, and fiber optics routing problems (Laywine and Mullen, 1998; Kumar, Russell, and Sundaram, 1999).

Our approach incorporates a new randomized $(1 - 1/e)$-approximation for QCP (Gomes, Regis, and Shmoys, 2003), the best approximation guarantee for this problem at the present time, a considerable improvement over the previously known performance guarantee for this problem, which was 0.5. We present experimental findings for a hybrid backtrack search method that tightly couples CSP propagation techniques with information provided by the randomized LP-based approximation for hard combinatorial instances of the QCP domain.

We compare our results with a pure CSP strategy and with a pure LP strategy. Our results show that a hybrid approach significantly improves over the pure strategies on hard instances. In our hybrid approach, the LP-based approximation provides global information about the values to assign to the CSP variables. In effect, the randomized LP rounding approximation provides powerful heuristic guidance to the CSP search, especially at the top of the backtrack search tree. With our hybrid CSP/LP strategy we were able to considerably improve the time performance of the pure CSP strategy on hard instances. Furthermore, the hybrid CSP/LP strategy could solve several instances of QCP that could not be solved by the pure CSP strategy.

Interestingly, and contrarily to the experience in other domains that combine linear constraints with a combinatorial component, we find that the role of the LP-based approximation in detecting infeasibility for pure combinatorial problems is not as important as its role as search guidance. In particular, deeper down in the search tree, the information obtainable via LP relaxations can be computed much faster via CSP techniques. This means that during that part of the search process, the hybrid strategy should be avoided. A key issue in making the hybrid strategy effective on purely combinatorial problems is to find the right balance between the amount of work spent in solving the LP relaxations and the time spent on the CSP search. Our approach also uses restart strategies in order to combat the heavy-tailed nature of combinatorial search. By using restart strategies we take advantage of any significant probability mass early on in the distribution, reducing the variance in runtime and the probability of failure of the search procedure, resulting in a more robust overall search method.

The structure of the paper is as follows. In the next section, we describe the Quasigroup Completion Problem (QCP). In section 3, we provide different formulations for the problem and, in section 4, we discuss approximations for QCP based on LP randomized rounding. In section 5, we present our hybrid CSP/LP randomized rounding backtrack search procedure and, in section 6, we provide empirical results.



	1	2	3
2		4	1
1	4		2
3		1	

4	1	2	3
2	3	4	1
1	4	3	2
3	2	1	4

Figure 1. Quasigroup completion problem of order 4, with 5 holes.

2. The quasigroup completion problem

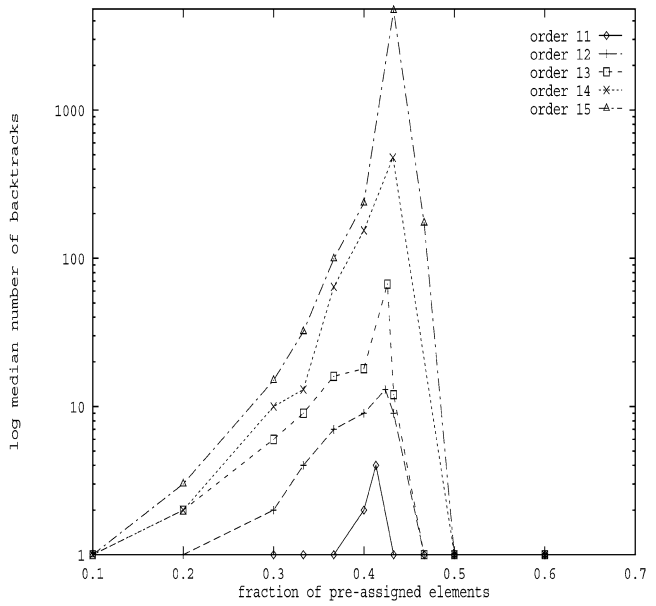
Given a set Q of n symbols (i.e., $|Q| = n$), a Latin square indexed by Q is an $n \times n$ matrix such that each of its rows and columns is a permutation of the n symbols in Q (Laywine and Mullen, 1998). n is called the *order* of the Latin square. From an algebraic point of view, a Latin square indexed by Q defines a very specific finite structure, a cancellative grupoid $(Q, \star)$, where $\star$ is a binary operation on Q and the Latin square is the multiplication table of $\star$. Such a grupoid is called a *quasigroup*. In the context of this paper, the key aspect about quasigroups that we are interested in is their equivalence to Latin squares, in the sense that the multiplication table of a quasigroup forms a Latin square and conversely, any given Latin square represents the multiplication table of a certain quasigroup. A *partial latin square PLS* is a partially filled n by n matrix such that no symbol occurs repeated in a row or a column. $PLS_{i,j} = k$ denotes that entry i, j of *PLS* has symbol k. We refer to the empty cells of the partial Latin square as *holes* and to the non-empty cells as *pre-assigned* cells. The number of holes in a *PLS* is denoted by h. The Quasigroup Completion Problem (QCP) (or Latin square completion problem)[1] is the problem of determining whether the h holes of the corresponding partial Latin square can be filled in such a way that we obtain a complete Latin square (i.e., a full multiplication table of a quasigroup) (see figure 1). QCP is NP-complete (Colbourn, 1984).

The structure implicit in QCP is similar to that found in real-world domains: indeed, many problems in scheduling and experimental design have a structure similar to the structure of QCP. A particularly interesting application that directly maps onto the QCP is the problem of assigning wavelengths to routes in fiber-optic networks, as performed by Latin routers (Kumar, Russell, and Sundaram, 1999). As the name suggests, Latin routers use the concept of Latin squares to capture the constraints required to achieve conflict-free routing: a given wavelength cannot be assigned to a given input port more than once; a given wavelength cannot be assigned to a given output port more than once.

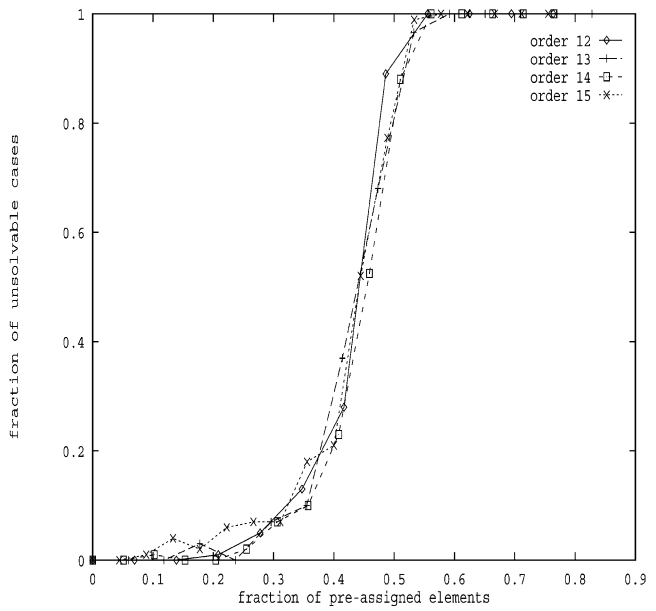
2.1. Phase transition phenomena in QCP

In recent years significant progress has been made in the study of typical case complexity, namely the study of so-called phase transition phenomena in computational models, 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-complete 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 and Huberman, 1996). For example, in the random Satisfiability problem, with fixed clause length (K-Sat, with $K \geqslant 3$), it has been empirically shown that the difficulty of problems depends on the ratio between number of clauses and number of variables (Kirkpatrick and Selman, 1994). Furthermore, it has also been empirically shown that the peak in complexity occurs at the phase transition, i.e., the region in which instances change from being almost all solvable to being almost all unsolvable. We note that a rigorous and formal characterization of phase transition phenomena of computational search problems is very challenging. Even though several researchers from different communities, namely theoretical computer science, mathematics, and physics are now working on the study of phase transition phenomena in computational methods, progress is very slow. For example, for the 3-Sat case, the existence of a phase transition phenomenon has been formally proved, but the exact ratio of clauses to variables at which it occurs has not been rigorously proved. While empirical results show that the phase transition phenomenon occurs when the ratio of clauses to variables is about 4.2, the theoretical results indicate that the phase transition phenomenon occurs between 3.4 and 4.6. There is a large body of recent work on the phase transitions observed in random distributions of constraint satisfaction problems, such as random Sat, random binary CSP, random graph coloring, and other problem domains. (For example, (Hogg and Huberman, 1996) contains a collection of recent papers in the area.)

We have identified a phase transition phenomenon for QCP (Gomes and Selman, 1997), a more structured domain than purely random problem domains such as random K-Sat. Interestingly, like in the random 3-Sat problem, the computationally hardest instances lie at the phase transition, the region in which problem instances switch from being almost all solvable ("under-constrained" region) to being almost all unsolvable ("over-constrained" region). Figure 2 shows the computational cost (median number of backtracks) and phase transition in solvability for solving QCP instances of different orders. In both of the plots displayed in the figure we vary the ratio of pre-assigned cells[2] along the horizontal axis. We vary the median number of backtracks for solution along the Y-axis of the top plot.[3] We vary the percentage of instances for which there is a valid completion of the partial Latin square along the vertical axis of the bottom panel. Each data point was obtained by running a complete backtrack search procedure on 100 instances for the specified ratio of pre-assigned cells. Even though all the instances are from QCP, which is an NP-complete problem, we clearly distinguish various regions of problem difficulty in the figure. In particular, both at low ratios and high ratios of preassigned colors the median solution cost is relatively small. In the under-constrained area, for example for levels of pre-assignment below 30%, the median number of backtracks for finding a solution is below 20 for all the orders of Latin squares shown in the figure. In the over-constrained area, for example for levels of pre-assignment greater than 50%, it is relatively easy for the backtrack search procedure to prove that the par-



(a)



(b)

Figure 2. (a) Computational cost of solving QCP instances (order 11–15). X-axis: fraction of pre-assigned cells; Y-axis: median number of backtracks for solution (log scale). (b) Phase transition in solvability for QCP instances (order 12–15). X-axis: fraction of pre-assigned cells; Y-axis: fraction of instances for which the partial Latin square could not be completed into a full Latin square. (Each data point was computed based on 100 instances.)

tial Latin squares cannot be completed into full Latin squares. The median number of backtracks for solution for instances with 50% pre-assigned colors is 1. In between the under-constrained and over-constrained area, the complexity peaks and, in fact, exhibits strong exponential growth. For example, for instances of order 15 in the critically constrained region, where the phase transition occurs, the median number of backtracks is greater than 4000.

We note that the location of the phase transition depends only on the structure of the particular domain and is not dependent on the particular search procedure: the transition is with respect to the solvability of the instances, a structural feature of the domain. For QCP, the transition is from a region in which for most instances the partial Latin square pattern can be completed to a full Latin square, to a region in which for most of the instances the partial pattern cannot be completed to a full Latin square. The computational cost complexity peaks in the phase transition region. While the location of the complexity cost peak is not sensitive to the search procedure in general, its magnitude, on the other hand, is highly correlated with the strength of the search method: the weaker the search method the higher the peak in computational cost.

There are only empirical results for the location of the phase transition for the quasigroup domain. Rigorous results are very difficult to obtain given the constraints that characterize this domain. For example, for a variation of QCP in which only feasible instances are considered, referred to as Quasigroup with Holes (QWH), we conjecture that an expression of the type $((1 - h)/n^p)$, with p constant, captures the location of the phase transition for the quasigroup domain, where h is the number of holes, and p is a scaling parameter. Based on empirical results p seems to be $\approx$1.55. However, given that for low orders of quasigroups $p = 2$ is a good approximation, for simplification we talk about proportion of holes (or preassigned colors) in terms of the total number of cells of the matrix, i.e., $(1 - h)/n^2$ (Achlioptas et al., 2000). The experiments reported in this paper were based on QWH instances (see section 6).

2.2. Heavy-tailed behavior in QCP

The runtime distributions of randomized backtrack search algorithms are highly nonstandard (Gomes et al., 2000). Technically speaking, these distributions are often *heavy-tailed*. Stated informally, when solving a computationally hard problem with a randomized search method, both very long runs and very short runs occur *much more frequently* than one might intuitively expect. For example, in practice, one can observe runs that take *only seconds* while other runs take *hours or days*, even when solving the same problem instance.

One of the clues that we are in the presence of a distribution with heavy-tails is the erratic behavior of its mean and/or variance. Figure 3 illustrates the "wandering sample mean" phenomenon of a randomized backtrack search algorithm running on a QCP instance. The figure displays the average cost (number of backtracks) of a randomized backtrack style search procedure calculated over an increasing number of runs, on the *same* QCP instance, an instance of order 11 with 64% of holes. Despite the fact that this

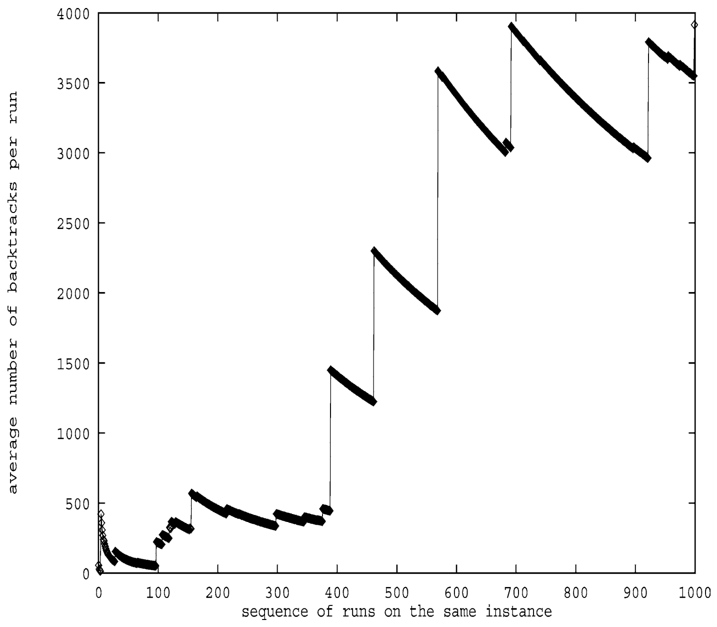


Figure 3. Erratic behavior of sample mean phenomenon. X-axis: sequence of runs on the same instance. Y-axis: average number of backtracks per run. The average of the number of backtracks until solution exhibits a very erratic behavior as we increase the number of runs of a randomized backtrack search algorithm on the same problem instance. After 200 runs, the mean is about 500 backtracks; after 600 runs, it is about 2000 backtracks; after 1000 runs, it is about 3500 backtracks. The phenomenon is due to the fact that the more runs we take into account, the more likely we hit some very long runs, which increase the overall average number of backtracks. This experiment consisted of using a randomized backtrack search algorithm (randomization only for tie breaking equal choices selected by the first-fail heuristic) on a QCP instance of order 11, with 64% of holes. Even though the randomized backtrack solver finds a solution in 50% of the cases using only one or zero backtracks (i.e., the median is 1), approximately 1% of the runs require more than 1,000,000 backtracks to find a solution.

instance is easy, in 50% of the runs a solution could be found with one or zero backtracks (median $= 1$), some runs take more than 10^6 backtracks.

To model the long tail behavior of our distributions, we consider distributions which asymptotically have tails of the Pareto–Lévy form, i.e., $\Pr\{X > x\} \sim Cx^{-\alpha}$, $x > 0$, with $\alpha > 0$. These are distributions whose tails have a power-law decay. The constant α is called the *index of stability* of the distribution. The distribution moments of order higher than α are infinite. For example, when $1 < \alpha < 2$, the distribution has a finite mean and infinite variance, and for $\alpha \leqslant 1$, both the mean and variance are not defined. 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 estimate the index of stability. As noted above, distributions of the Pareto–Lévy form have tails with power-law decay. Therefore, the log–log plot of the tail of the distribution should show an approximate linear decrease, as shown in figure 4. The visual check of the linearity of the plot can be confirmed by calculating the maximum

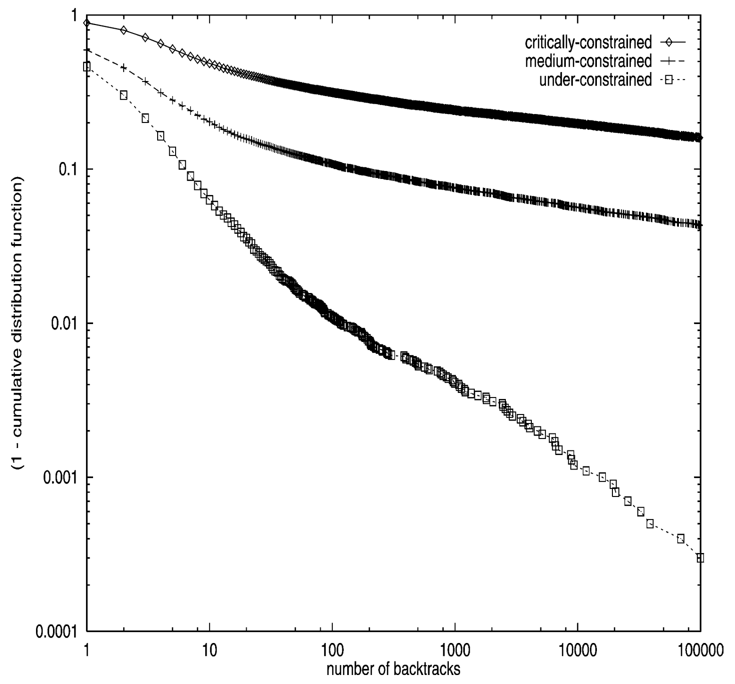


Figure 4. Heavy-tailed behavior for three different QCP instances: the top curve corresponds to a critically constrained instance; the middle curve corresponds to a medium-constrained instance; and the bottom curve corresponds to an under-cosntrained instance. X-axis: number of backtracks (log scale). Y-axis: the complement-to-one of the cumulative distribution (log scale) (i.e., the tail or the survival function of the distribution, $\Pr\{X > x\} = 1 - F(x)$, where $F(x)$ is the cumulative distribution). The linear behavior of the curves in log–log scale indicates heavy-tailed behavior. For example, the lower curve shows that ≈90% of the runs finish in less than 10 backtracks; ≈0.1% of runs take more than 10,000 backtracks; some of the runs take even more than 100,000 backtracks. Each distribution is based on 10,000 runs of the backtrack search method. For the under-constrained instance (bottom curve) the estimate of α is 0.466 ± 0.009. Given that $\alpha < 1$ and the data were obtained without censorship, the data are consistent with the hypothesis of infinite mean and infinite variance.

likelihood estimates of the indices of stability (the values of α), using a variation of the Hill estimator (Hill, 1975) modified to take into account data truncation of extreme outliers (Gomes et al., 2000). Because $\alpha < 1$ for all the distributions displayed in figure 4, in all those cases the data are consistent with the hypothesis of infinite mean and infinite variance.[4]

The formal explanation for heavy-tailed behavior comes from the fact that there is a non zero probability of entering a subtree of exponential size that has no solutions (Chen, Gomes, and Selman, 2001). The phenomenon of heavy-tailed distributions suggests that a sequence of "short" runs instead of a single long run may be a more effective use of our computational resources. As a direct practical consequence of the heavy-tailed behavior of cost distributions, randomized *restarts* of search procedures can dramatically reduce the variance in the search behavior. In fact, restarts eliminate heavy-tail behavior (Gomes et al., 2000).

3. Problem formulations

3.1. CSP formulation

Given a partial latin square of order n, *PLS*, the Latin square completion problem can be expressed as a CSP (Gomes and Selman, 1997):

$$\begin{array}{ll} x_{i,j} \in \{1, \ldots, N\} & \forall i, j, \\ x_{i,j} = k & \forall i, j \text{ such that } PLS_{ij} = k, \\ \texttt{alldiff}\,(x_{i,1}, x_{i,2}, \ldots, x_{i,n}), & i = 1, 2, \ldots, n, \\ \texttt{alldiff}\,(x_{1,j}, x_{2,j}, \ldots, x_{n,j}), & j = 1, 2, \ldots, n. \end{array}$$

$x_{i,j}$ denotes the symbol in cell i, j, and the statement "$PLS_{ij} = k$" denotes that symbol k is pre-assigned to cell i, j.

The alldiff constraint states that all the variables involved in the constraint have to have different values. It has been shown that a CSP approach solves QCP instances up to order about 33 relatively well (Gomes and Selman, 1997; Shaw, Stergiou, and Walsh, 1998; Achlioptas et al., 2000). However, given the exponential increase in search cost in the phase transition area. critically constrained instances of order greater than 33 are beyond the reach of pure CSP solvers.

3.2. Assignment formulation

Given a partial Latin square of order n, *PLS*, the Latin square completion problem can be expressed as an integer program (Kumar, Russell, and Sundaram, 1999):

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} x_{i,j,k}$$

subject to

$$\begin{array}{ll} \displaystyle\sum_{i=1}^{n} x_{i,j,k} \leqslant 1, & \forall j, k, \\ \displaystyle\sum_{j=1}^{n} x_{i,j,k} \leqslant 1, & \forall i, k, \\ \displaystyle\sum_{k=1}^{n} x_{i,j,k} \leqslant 1, & \forall i, j, \\ x_{i,j,k} = 1 & \forall i, j, k \text{ such that } PLS_{ij} = k, \\ x_{i,j,k} \in \{0, 1\} & \forall i, j, k, \\ & i, j, k = 1, 2, \ldots, n. \end{array}$$

If *PLS* is completable, the optimal value of this integer program is n^2, i.e., the total number of cells in the $n \times n$ Latin square. Kumar, Russell, and Sundaram (1997, 1999) considered the design of approximation algorithms for this optimization variant of the problem based on first solving the linear programming relaxation of this integer programming formulation; that is, the conditions $x_{i,j,k} \in \{0, 1\}$ above are replaced by $x_{i,j,k} \in [0, 1]$. Their algorithm repeatedly solves this linear programming relaxation, focuses on the variable closest to 1 (among those not set to 1 by the *PLS* conditions), and sets that variable to 1; this iterates until all variables are set. This algorithm is shown to be a $(1/3)$-approximation algorithm; that is, if *PLS* is completable, then it manages to find an extension that fills at least $h/3$ holes. Kumar, Russell, and Sundaram also provide a more sophisticated algorithm in which the colors are considered in turn; in the iteration corresponding to color k, the algorithm finds the extension (of at most n cells) for which the linear programming relaxation places the greatest total weight. This algorithm is shown to be a $(1/2)$-approximation algorithm; that is, if *PLS* is completable, then the algorithm computes an extension that fills at least $h/2$ holes. In the experimental evaluation of their algorithms, Kumar, Russell, and Sundaram solve problems up to order 9.

3.3. Packing formulation

Alternate integer programming formulations of this problem can also be considered. The *packing formulation* is one such formulation for which the linear programming relaxation produces stronger lower bounds. For the given *PLS* input, consider one color k. If *PLS* is completable, then there must be an extension of this solution with respect to this one color; that is, there is a set of cells (i, j) that can each be colored with color k so that there is exactly one cell colored with color k in every row and column. We shall call one such collection of cells a *compatible matching* for color k. Furthermore, any subset of a compatible matching shall be called a *compatible partial matching*; let $\mathcal{M}_k$ denote the family of all compatible partial matchings for color k.

With this notation in mind, then we can generate the following integer programming formulation by introducing one variable $y_{k,M}$ for each compatible partial matching M in $\mathcal{M}_k$:

$$\max \sum_{k=1}^{n} \sum_{M \in \mathcal{M}_k} |M| y_{k,M}$$

subject to

$$\begin{aligned} &\sum_{M \in \mathcal{M}_k} y_{k,M} = 1, && \forall k, \\ &\sum_{k=1}^{n} \sum_{M \in \mathcal{M}_k : (i,j) \in M} y_{k,M} \leqslant 1, && \forall i, j, \\ &y_{k,M} \in \{0, 1\}, && \forall k, M. \end{aligned}$$

Once again, we can consider the linear programming relaxation of this formulation, in which the binary constraints are relaxed to be in the interval $[0, 1]$. It is significant to note that, for any feasible solution y to this linear programming relaxation, one can

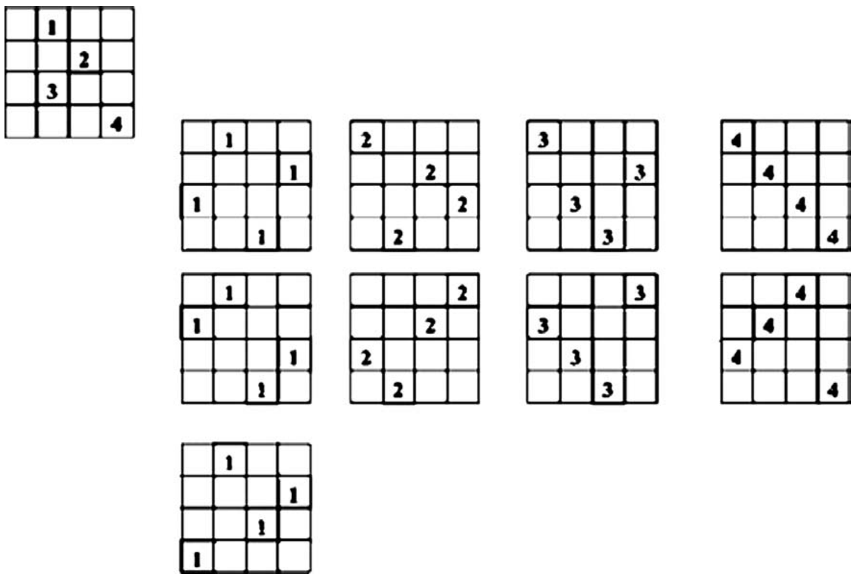


Figure 5. Families of compatible matchings for the partial Latin square in the left upper corner. For example, the family of compatible matchings for symbol 1 has three compatible matchings.

generate a corresponding feasible solution x to the assignment formulation, by simply computing $x_{i,j,k} = \sum_{M \in \mathcal{M}_k : (i,j) \in M} y_{k,M}$. This construction implies that the value of the linear programming relaxation of the assignment formulation (which provides an upper bound on the desired integer programming formulation) is at least the bound implied by the LP relaxation of the packing formulation; that is, the packing formulation provides a tighter upper bound. However, note that the size of this formulation is exponential in n. Despite this difficulty, one may apply the ellipsoid algorithm (via its dual) to solve the packing LP relaxation in polynomial time (Grotschel, Lovász, and Schrijver, 1993), or apply more specialized LP techniques designed to give fully polynomial approximation schemes for such packing-type linear programs (e.g., Plotkin, Shmoys, and Tardos, 1995). In practice, one can simply apply column generation techniques to compute an optimal solution relatively efficiently.

4. Approximations based on randomized rounding

One important area of recent research has been the design of approximation algorithms. Approximation algorithms run in polynomial time, with the additional guarantee of producing solutions that are close to the optimal solution. The notion of being *close to the optimal solution* is usually specified with a performance guarantee parameter α. An α-approximation algorithm is a polynomial-time algorithm that finds a feasible solution of objective function value within a factor α of the optimal solution. We consider approximation algorithms in which good solutions are computed for an integer programming problem with variables constrained to be 0 or 1, by solving its linear programming relaxation, and (appropriately) interpreting the resulting fractional solution as providing a probability distribution over which to set the variables to 1 (see, e.g., (Motwani, Naor, and Raghavan, 1997)).

Consider the generic integer program $\max cz$ subject to $Az = b$, $z \in \{0, 1\}^N$, and solve its linear relaxation to obtain z^*. If each variable z_j is then set to 1 with probability ${z_j}^*$, then the expected value of the resulting integer solution is equal to the LP optimal value, and, for each constraint, the expected value of the left-hand side is equal to the right-hand side. Of course, this does not mean that the resulting solution is feasible, but it provides a powerful intuition for why such a *randomized rounding* is a useful algorithmic tool (see, e.g., (Motwani, Naor, and Raghavan, 1997)).

This approach has led to striking results in a number of settings. For example, Goemans and Willianson (1994) have given a (3/4)-approximation algorithm based on randomized rounding for the problem of satisfying the maximum number of clauses for a boolean formula in conjunctive normal form. This algorithm outputs the better solution found by two randomized rounding procedures, one that uses a fair coin to independently set the variables, and another that randomly rounds based on the optimal solution to a natural linear programming relaxation.

4.1. Assignment formulation

The assignment formulation can be used as the basis for a randomized rounding procedure in a variety of ways. Let x^* denote an optimal solution to the linear programming relaxation of this integer program. For any randomized procedure in which the probability that cell (i, j) is colored k is equal to x^*_{ijk}, then we know that, in expectation, each row i has at most one element of each color k, each column j has at most one element of each color k, and each cell (i, j) is assigned at most one color k. For example, if $x^*_{ijk} = 0.8$, we will assign color k to cell (i, j) with probability 0.8. However, having these each hold "in expectation" is quite different than expecting that all of them will hold simultaneously, which is extremely unlikely.

4.2. Packing formulation

In contrast to the situation for the assignment formulation, there is an easy theoretical justification for the randomized rounding of the fractional optimal solution, as we proposed in (Gomes, Regis, and Shmoys, 2003). Let y^* be an optimal solution to the LP relaxation of the packing formulation. Rather than the generic randomized rounding mentioned above, instead, for each color k choose some compatible partial matching M with probability $y^*_{k,M}$ (so that some matching is therefore selected for each color). These selections are done as independent random events. This independence implies there might be some cell (i, j) included in the matching selected for two distinct colors. However, the constraints in the linear program imply that the expected number of matchings in which a cell is included is at most one. For each cell in which a overlap occurs, we arbitrarily select a color from among the colors involved in the overlap. It is easy to see that the result is an extension of the original *PLS*. Furthermore, this procedure yields a $(1 - 1/\mathrm{e})$-approximation for the partial Latin square extension problem, the optimization version of the (decision) problem of completing a Latin square.

Theorem 4.1. Randomized rounding based on the packing formulation yields a $(1 - 1/\mathrm{e})$-approximation algorithm for the partial Latin square extension problem.

Proof sketch. Let y^* be an optimal solution to the LP relaxation of the packing formulation. The probability of cell (i, j) being colored by color k is given by $x^*_{i,j,k} = \sum_{M\in\mathcal{M}_k:(i,j)\in M} y^*_{k,M}$. So, the probability of a cell not being colored, corresponds to the probability of a cell not being covered by any matching selected by the randomizing rounding procedure, i.e., $\prod_k(1 - x^*_{ijk})$. This expression is maximized when all the x^*_{ijk} take the same value, i.e., when $x^*_{ijk} = 1/n$. Therefore, $\prod_k(1 - x^*_i jk) \leqslant (1 - 1/n)^n \leqslant 1/\mathrm{e}$, and the expected number of uncolored cells is at most Z^*/e, where Z^* is the optimal solution, i.e., the maximum number of holes that can be filled in the problem of extending the partial Latin square. Therefore, at least $(1 - 1/\mathrm{e})Z^*$ holes are expected to be filled by this technique.

Note that if *PLS* is completable, and hence the linear programming relaxation satisfies the inequality constraints with equality (and hence $|M| = n$ whenever $y_{k,M} > 0$), then the expected number of cells not covered by any matching is is at most h/e; that is, at least $(1-1/\mathrm{e})h$ holes can expected to be filled by this technique. (See (Gomes, Regis, and Shmoys, 2003) for details.) □

4.3. Assignment vs. packing formulation

Although the LP relaxation of the packing formulation appears to provide a stronger lower bound, in fact, the bound is identical to the one given by the LP relaxation of the assignment formulation. To see this, one needs only the fact that the extreme points of each polytope

$$P_k = \left\{x\colon \sum_{i=1}^{n} x_{ijk} \leqslant 1,\ j = 1, \ldots, n, \sum_{j=1}^{n} x_{ijk} \leqslant 1,\ i = 1, \ldots, n,\ x \geqslant 0\right\},$$

for each $k = 1, \ldots, n$ are integer, which is a direct consequence of the Birkhoff–von Neumann Theorem (von Neumann, 1953). Furthermore, these extreme points correspond to matchings, i.e., a collection of cells that can receive the same color.

Hence, given the optimal solution to the assignment relaxation, if we fix the color k, and consider the values x_{ijk}, since this is a vector in P_k, we can write it as a convex combination of extreme points, i.e., matchings, and hence obtain a feasible solution to the packing formulation of the same objective function value. Hence, the optimal value of the packing relaxation is at most the value of the assignment relaxation, i.e., the optimal values are equal.

Finally, it is possible to compute the convex combination of the matchings efficiently; one way to view this is as a special case of preemptive open shop scheduling and hence one can use a result of (Lawler and Labetoulle, 1978) to produce, in polynomial time, a polynomial number of matchings, along with corresponding multipliers, which specify the desired convex combination. Hence, the most natural view of the algorithm is to solve the assignment relaxation, compute the decomposition into matchings, and then perform randomized rounding to compute the partial completion.

5. Hybrid CSP/LP randomized rounding backtrack search

As mentioned in the Introduction, in recent years there has been an increasing interest in hybrid approaches for combinatorial optimization (see, e.g., (Caseau and Laburthe, 1995; Darby-Dowman et al., 1997; Bockmayr and Kasper, 1998; Hooker et al., 1999; Focacci, Lodi, and Milano, 2000)). This is especially true for problems characterized by a mixture of combinatorial and linear constraints. The general approach underlying hybrid approaches is to combine a constraint programming model with a mixed integer programming formulation. The two models are linked and domain reductions and/or infeasibility information discovered in one of the models is passed to the other, and vice-versa.

Our approach follows the general philosophy of hybrid methods, maintaining two linked models that share information discovered by the corresponding inference methods. However, a key difference in our approach is the use of a randomized rounding LP approximation, with performance guarantees, to go from the relaxed LP solution to the original problem, and based on that information, set variable/values in the backtrack search procedure. Randomization is a powerful tool, especially when combined with restart strategies. Our randomized search method increases the robustness of standard backtrack search and extends the reach of such methods considerably, as we will demonstrate on our quasigroup (Latin square) completion problem, a *pure* combinatorial problem domain. Our technique is general and therefore should apply to a range of combinatorial problems.

We now describe our complete randomized backtrack search algorithm, applied to the quasigroup (Latin square) completion problem.

The algorithm maintains two different formulations of the quasigroup completion problem: a CSP formulation, as described in section 3.1, and a relaxation of the LP formulation described in section 3.2. The hybrid nature of the algorithm results from the combination of strategies for variable and value assignment, based on the LP approximation and on the CSP formulation, and propagation, based on the two underlying models.

Algorithm 5.1. Given a problem instance I, a CSP model, an LP model, a parameter cutoff and a function Δ(cutoff), and parameters %LP and InterleaveLP:

```
Initialization:
  Instantiate variables in the CSP model; pre-process the CSP model.
  Instantiate the variables in the LP model based on the pre-processed CSP model.
  vars $\leftarrow$ (number of variables in CSP model after propagation).
Repeat until proving optimality.
  cutoff $\leftarrow$ (cutoff $+ \Delta($cutoff$)), \Delta($cutoff$) \geqslant 0$.
  numVarAssignments $\leftarrow 0$.
  Perform backtrack search until proving optimality or cutoff reached:
    While numVarAssignments $\leqslant$ (%LP $\times$ vars$/100)$):
      Select variable/value based on the LP (randomized rounding);
      Perform CSP inference; update LP model based on CSP inference.
      Update numVarAssignments.
      Re-solve LP with frequency according to parameter InterleaveLP;
    Select variable/value based on the CSP model;
    Perform CSP inference.
```

The algorithm is initialized by instantiating the variables in the CSP model and pre-processing the CSP model. In this pre-processing phase inference is performed based only on the CSP model. In CSP models, inference corresponds to variable domain reduction and propagation. Variable domain reduction (or filtering) corresponds to removing values from the variable domains that provably do not belong to an optimal solution. In the pre-processing phase domain reduction is performed for each constraint and for all the variables involved in the constraints. Propagation occurs since performing domain reduction on the variables associated with one constraint can lead to domain reduction in the domains of other variables, involved in other constraints. When a variable domain is modified, all the constraints involving this variable are considered for domain reduction, which leads to the modification of the domains of other variables, which in turn may involve other constraints, and so on, until reaching a fixed point. The CSP model is implemented in Ilog/Solver (Ilog, 2001b). For the domain reduction/propagation of the ALLDIFF constraint we use the extended version provided by (Ilog, 2001b) (see (Regin, 1994) for details on the ALLDIFF constraint). The updated domain values of the variables in the CSP model (after the pre-processing phase) are then used to instantiate the variables in the LP model. We solve the LP model using Ilog/Cplex Barrier (Ilog, 2001a).

As we will see from our experiments below, the LP provides valuable search guidance and pruning information for the CSP search. However, since solving the LP model is relatively expensive compared to the inference steps in the CSP model, we have to carefully manage the time spent on solving the LP model. The LP effort is controlled by two parameters, as explained below.

In the initial phase of the backtrack search procedure, corresponding to the "top" of the search tree, variable and value selection are based on the LP approximation. After each variable/value assignment based on the LP randomized approximation, full propagation is performed on the CSP model. The percentage of variables set in this initial phase corresponds to the %LP applied to the total number of variables in the original CSP (after the initial propagation). After this initial phase, variable and value settings are based purely on the CSP model. Note that deeper down in the search tree, the LP formulation continues to provide information on variable settings and also on inconsistent tree nodes. However, we have found that, at lower levels of the search tree, the CSP model can infer inconsistent variable values and detect general inconsistency much more efficiently than solving the relaxation of the LP. In other words, solving the LP relaxation is not cost-effective at the lower levels of the search tree since the same, or even higher, level of pruning information can be obtained by using only CSP pruning techniques, that are less expensive than solving the LP relaxation.

Ideally, in order to increase the accuracy of the variable assignments based on the LP-rounding approximation, one would like to update and re-solve the LP model after each variable setting. However, in practice, this is too expensive. We therefore introduce a parameter, Interleave-LP, which determines the frequency with which the LP model is updated and re-solved. In our experiments, we found that updating the LP model after every five variable settings (Interleave-LP $= 5$) is a good compromise.

In the search guided by the CSP model, we use a variant of the Brelaz heuristic (Brelaz, 1979; Gomes and Selman, 1997) for the variable and value selections. The Brelaz heuristic is a popular extension of the First-fail heuristic, which was originally introduced for graph coloring procedures. In the First-fail heuristic, the next variable to branch on is the one with the smallest remaining domain, i.e., the search procedure chooses to branch on the variable with the fewest possible options left to explore, therefore leading to the smallest branching factor. In case of ties, the standard approach is to break ties using lexicographical order. The Brelaz heuristic specifies a way for breaking ties in the First-fail rule: If two variables have equally small remaining domains, the Brelaz heuristic chooses the variable that shares constraints with the largest number of the remaining unassigned variables. We used a natural variation on this tie-breaking rule, what we called "the reverse-Brelaz" heuristic, in which preference is given to the variable that shares constraints with the *smallest* number of unassigned variables. Any remaining ties after the reverse-Brelaz heuristic are resolved randomly.

Backtracking can occur as a result of an inconsistency detected either by the CSP model or the LP relaxation. It is interesting to note that backtracking based on inconsistencies detected by the LP model occurs rather frequently at the top of our search tree. This means that the LP does indeed uncover global information not easily obtained via CSP propagation, which is a more local inference process. Of course, as noted before, lower down in the search tree, using the LP for pruning becomes ineffective since CSP propagation can uncover the same information with only a few additional backtracks.

In this setting, we are effectively using the LP as heuristic guidance, using a randomized rounding approach inspired by the rounding schemes used in approximation algorithms. For the empirical results reported in this paper we used the LP rounding strategy directly based on the assignment formulation.[5] We first rank the variables according to their LP values (i.e., variables with LP values closest to 1 are ranked near the top). We then select the highest ranked variable and set its value to 1 (i.e., set the color of the corresponding cell) with a probability p given by its LP value. With probability $1 - p$, we randomly select a color for the cell from the colors still allowed according to the CSP model. After each variable setting, we perform CSP propagation. The CSP propagation will set some of the variables on our ranked variable list. We then consider the next highest ranked variable that is not yet assigned. A total of Interleave-LP variables is assigned this way, before we update and re-solve the LP.

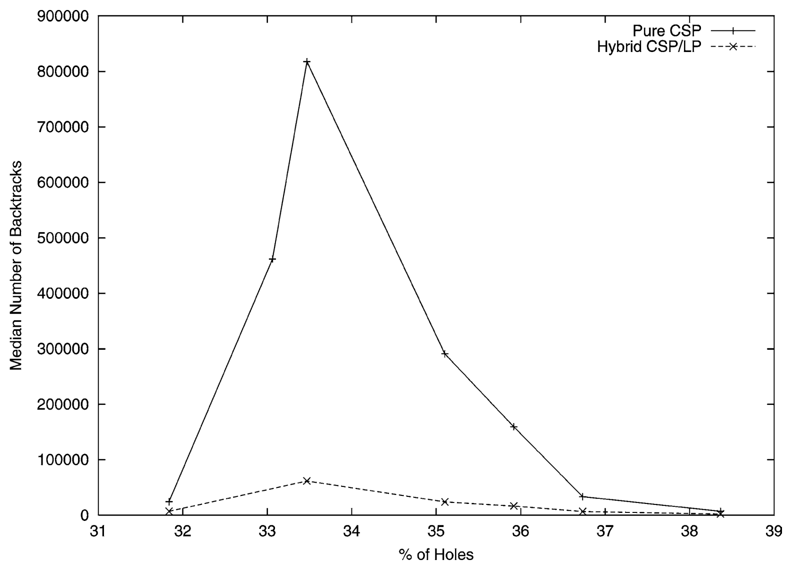
Finally, we use a cutoff parameter to control our backtrack search. As mentioned in section 2, backtrack search methods are characterized by heavy-tailed behavior. That is, a backtrack search is quite likely to encounter extremely long runs. To avoid getting stuck in such unproductive runs, we use a cutoff parameter. This parameter defines the number of backtracks after which the search is restarted, at the beginning of the search tree, with a different random seed. Note that in order to maintain the completeness of the algorithm we just have to increase the cutoff, according to some non-negative function, $\Delta(\text{cutoff})$. In practice, we increase the cutoff linearly every certain number of restarts. In the limit, we run the algorithm without a cutoff. Restart strategies have also been shown very powerful to boost performance of complete methods for Boolean Satisfiability (see, e.g., (Moskewicz et al., 2001; Baptista, Lynce, and Marques-Silva, 2001)).

6. Empirical results

To investigate our hypothesis that the LP-based approximation can provide useful search guidance, we focused our empirical evaluation on solvable instances. To do so, we used a variant of the QCP problem, in which we generate instances for which we are guaranteed that a solution exists. To obtain such instances, we start with a randomly generated complete Latin square and uncolor a fraction of cells (randomly selected). The random complete Latin square is generated using a Markov chain Monte Carlo shuffling process (Jacobson and Matthews, 1996). The task again is to find a coloring for the empty cells that completes the Latin square. We refer to this problem as the "quasigroup with holes" (QWH) problem.[6] We can again finely tune the complexity of the completion task by varying the fraction of the uncolored cells (Achlioptas et al., 2000).

In figure 6, we compare the performance of our hybrid CSP/LP strategy against the pure CSP strategy. For the hybrid CSP/LP strategy we set %LP $=$ 10 and Interleave-LP $=$ 5. Each data point was obtained by running the randomized search procedures on 100 different instances, with a cutoff of 10^6, and computing the median in number of backtracks (upper panel) and total runtime (lower panel). From the figure, we again see the easy-hard-easy pattern, both in the hybrid CSP/LP and the pure CSP strategy. Moreover, the hybrid CSP/LP strategy significantly outperforms the pure CSP strategy, both in terms of the number of backtracks and total runtime. The relative payoff of our hybrid strategy is largest for the hardest problem instances (about 33.6% holes).

We now consider more detailed performance data on three hard problem instances. In table 1 we show the performance of the CSP/LP strategy and the pure CSP strategy on an instance of order 35 with 405 holes (33% holes). (This instance is medium hard



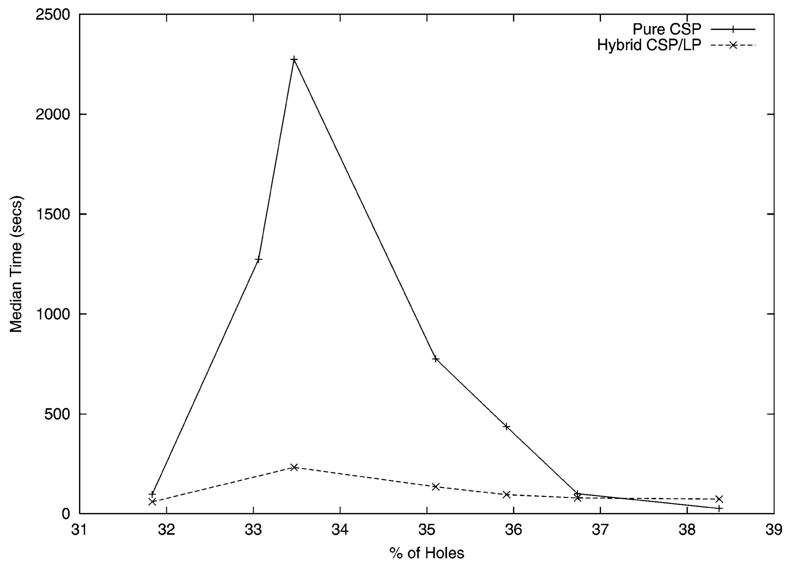


Figure 6. Median runtime (secs) for QWH instances, in the *critically constrained area* (order 35; 100 instances per data point).

– somewhat before the phase transition region.) The pure CSP strategy can solve this instance using a high cutoff of 10^6, but only in 6% of the runs. On the other hand, the CSP/LP strategy is much more effective. In fact, even with only %LP $= 1$, we can solve the instance in 42% of the runs. With %LP $\geqslant 10$, we solve the instance on each run. Looking at the overall runtime as a function of %LP, we see that at some point further use of LP relaxations becomes counterproductive. The best performance is obtained with %LP about 20.

Table 1
Hybrid CSP/LP search on an instance of order 35 with 405 holes.

%LP	Cutoff	Num. runs	% succ. runs	Median backtracks	Median time
0	10^6	100	6%	474049	1312.58
1	10^6	100	42%	589438	1992.08
5	10^6	100	90%	188582	615.16
10	10^6	100	100%	26209	114.35
15	10^6	100	100%	22615	116.29
20	10^6	100	100%	17203	112.64
25	10^6	100	100%	21489	158.07
30	10^6	100	100%	24139	179.37
50	10^6	100	100%	19325	262.67
75	10^6	100	100%	17458	379.68

Table 2
Instance of order 40, with 528 holes.

%LP	Cutoff	Num. runs	% succ. runs	Median backtracks	Median time
0	10^5	100	0%	N.A.	N.A.
10	10^5	100	1%	48387	245.54
25	10^5	100	37%	17382	215.69
50	10^5	100	47%	21643	422.59
0	10^6	100	0%	N.A.	N.A.
10	10^6	100	8%	355362	1488.08
25	10^6	100	64%	123739	574.68
50	10^6	100	65.3%	128306	757.55

In tables 2 and 3, we consider, respectively, a critically constrained instance of QWH of order 40, with 528 holes, and a medium constrained instance of QWH of order 40, with 544 holes. We were unable to solve these instances with a pure CSP strategy using a cutoff of 10^5 (100 runs) and a cutoff of 10^6 (100 runs). Both instances can be solved with the hybrid CSP/LP strategy. From the median overall runtime, we see that the best performance is obtained for %LP about 20–25. For example, in the case of the instance of order 35 with 405 holes, the median time for solution decreases as we increase the value of %LP, achieving the best performance for %LP $= 20$. For %LP > 20, the median time increases. In the case of the instances of order 40 (528 holes and 544 holes) the best performance is achieved when %LP $= 25$.

We note that pure integer programming based methods perform very poorly on this highly combinatorial domain, in comparison with CSP methods. In fact, hard instances of QCP/QWH are out of reach of a pure integer programming strategy (i.e., no interleaved CSP propagation): the pruning power provided by the CSP component is critical in this highly combinatorial domain. Our hybrid method further extends the range of

Table 3
Instance of order 40, with 544 holes.

%LP	Cutoff	Num. runs	% succ. runs	Median backtracks	Median time
0	10^5	100	0%	N.A.	N.A.
10	10^5	100	1%	41771	264.96
25	10^5	100	34%	31386	287.72
50	10^5	100	38%	13266	395.31
0	10^6	100	0%	N.A.	N.A.
10	10^6	100	5%	167897	813.58
25	10^6	100	53%	110787	560.56
50	10^6	100	92%	75234	648.87

QCP/QWH problems we can solve with CSP based methods. The hybrid method is particularly suitable for hard instances, out of reach of the pure CSP strategy. In fact, for instances that are easily solved by the pure CSP strategy, the hybrid strategy is less efficient than the pure CSP strategy. The payoff of our hybrid strategy is largest for hard problem instances, out of reach of the pure CSP strategy, The hybrid strategy allows us to improve on the time performance of the pure CSP strategy and reliably solve larger instances, out of reach of the pure CSP strategy, up to order 40–45. We also solved several hard instances of order 50. On our very hardest problem, we had to increase %LP to about 50%. So, apparently, more guidance was required from the LP relaxation.

7. Conclusions

We have demonstrated the use of approximations and randomization to boost CSP methods on hard purely combinatorial problems. Our approach involves an LP-based randomized rounding strategy inspired by recent rounding methods used in approximation algorithms. In this setting, the LP randomized rounding provides powerful guidance in the CSP search. Randomization and restarts in the backtrack process are needed to make the overall strategy robust and to recover from possible early branching mistakes. Essential to our approach is a tight coupling of a CSP and a LP model of the problem, that we simultaneously maintain. The local nature and high efficiency of the CSP propagation methods enable us to call such methods frequently. In particular, CSP propagation is performed after each variable assignment. By frequently updating and resolving the LP model, our LP-based approximation rounding decisions stay accurate during the search process. The continuous interleaving of CSP propagation and LP guidance using a randomized rounding approximation are key features of our approach. Also, we carefully control the amount of time spent in solving the LP relaxations, by restricting this process to the top of the backtrack search tree and not solving the LP at every node of the search tree.

In experiments, we were able to significantly extend the reach of CSP and LP techniques for solving instances of the quasigroup (Latin square) completion problem. Our technique is general and therefore holds promise for a range of combinatorial problems. We believe there is still room for further improvement. For example, using different CSP formulations and LP-based approximations. Furthermore, we believe that our analysis of the packing formulation is, most likely, not tight. A possible avenue for improvement is by means of a primal–dual approach. We hope this work provides researchers in the Constraint Programming and Operations Research communities with new insights about the power of randomization and approximation algorithms for boosting performance of complete backtrack search methods.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. This research was partially supported by AFOSR grants F49620-01-1-0076 (Intelligent Information Systems Institute) and F49620-01-1-0361 (MURI) and NSF grant (CCR-9912422).

Notes

1. The designation of quasigroup completion problem was inspired by the work done by the theorem proving community on the study of quasigroups as highly structured combinatorial problems. For example, the question of the existence and non-existence of certain quasigroups with intricate mathematical properties gives rise to some of the most challenging search problems (Slaney, Fujita, and Stickel, 1995). For simplicity, in the remaining of the paper, we will use quasigroup and Latin square and partial quasigroup completion problem and partial Latin square completion problem interchangeably.
2. Note that the ratio of pre-assigned cells corresponds to the complement of the ratio of holes.
3. Note that we use the median instead of the average to characterize the computational cost due to the fact that the mean is not stable when computed over different ensembles of instances. This aspect will be made clearer below.
4. In fact, the computational cost of any complete backtrack algorithm has a finite upper-bound. However, because the upper-bound is exponential in the size of the problem, from a computational point of view it can be treated as infinite in the probabilistic model.
5. We are currently implementing the strategy that maps the assignment formulation onto the packing formulation, as described in section 4.3, as well as a column generation approach for the packing formulation. It is not clear whether such approaches will improve considerably upon the results obtained using the LP rounding strategy directly based on the assignment formulation.
6. The code for this generator is available by contacting Carla Gomes (gomes@cs.cornell.edu).

References

Achlioptas, D., C. Gomes, H. Kautz, and B. Selman. (2000). "Generating Satisfiable Instances." In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pp. 256–261. New Providence, RI: AAAI Press.

Baptista, L., I. Lynce, and J. Marques-Silva. (2001). "Complete Restart Strategies for Satisfiability." In *Proceedings of the Workshop on Stochastic Search Algorithms, Seventeenth International Joint Conference on Artificial Intelligence (IJCAI01)*, Seattle, WA.

Bockmayr, A. and T. Kasper. (1998). "Branch and Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming." *INFORMS Journal on Computing* 10, 287–300.

Brelaz, D. (1979). "New Methods to Color the Vertices of a Graph." *Communications of the ACM* 22(4), 251–256.

Caseau, Y. and F. Laburthe. (1995). "Improving Branch and Bound for Jobshop Scheduling with Constraint Propagation." In *Proceedings of the 8th Franco-Japanese and 8th Franco-Chinese Conference on Combinatorics and Computer Science*, Brest, France, Lecture Notes in Computer Science, Vol. 1120. New York: Springer.

Chen, H., C. Gomes, and B. Selman. (2001). "Formal Models of Heavy-Tailed Behavior in Combinatorial Search." In *Proceedings of 7th International Conference on the Principles and Practice of Constraint Programming (CP-2001)*, Paphos, Cyprus, Lecture Notes in Computer Science, Vol. 2239, pp. 408–422. New York: Springer.

Chudak, F. and D. Shmoys. (1999). "Improved Approximation Algorithms for the Uncapacitated Facility Location Problem." Submitted for publication. Preliminary version of this paper (with the same title) appeared in *Proceedings of the Sixth Conference on Integer Programming and Combinatorial Optimization*, 1999.

Colbourn, C. (1984). "The Complexity of Completing Partial Latin Squares." *Discrete Applied Mathematics* 8, 25–30.

Darby-Dowman, K., J. Little, G. Mitra, and M. Zaffalon. (1997). "Constraint Logic Programming and Integer Programming Approaches and Their Collaboration in Solving an Assignment Scheduling Problem." *Constraints* 1(3), 245–264.

Focacci, F., A. Lodi, and M. Milano. (1999a). "Cost-Based Domain Filtering." In *Proceedings of 5th International Conference on the Principles and Practice of Constraint Programming (CP-1999)*, Alexandria, VA, Lecture Notes in Computer Science, Vol. 1713, pp. 189–203. New York: Springer.

Focacci, F., A. Lodi, and M. Milano. (1999b). "Solving Tsp with Time Windows with Constraints." In *Proceedings of the 16th Conference on Logic Programming, (ICLP 99)*, Las Cruces, NM, pp. 515–529.

Focacci, F., A. Lodi, and M. Milano. (2000). "Cutting Planes in Constraint Programming: An Hybrid Approach." In *Proceedings of 6th International Conference on the Principles and Practice of Constraint Programming (CP-2000)*, Singapore, Lecture Notes in Computer Science, Vol. 1894, pp. 187–201. New York: Springer.

Goemans, M. and D.P. Willianson. (1994). "0.878-Approximation Algorithms for Max-Cut and Max-Sat." In *Proceedings of the 26th Annual ACM–SIAM Symposium on Theory of Computing*, San Diego, CA, pp. 422–431.

Gomes, C. (ed.). (2000–2001). *The Knowledge Engineering Review* 15(1) and 16(1), Special Issue on the Integration of Artificial Intelligence and Operations Research Techniques.

Gomes, C., R. Regis, and D. Shmoys. (2003). "An Improved Approximation Algorithm for the Partial Latin Square Extension Problem." In *Proceedings of the Fourteenth Annual ACM–SIAM Symposium on Discrete Algorithms (SODA-03)*, Baltimore, MD, pp. 832–833.

Gomes, C. and B. Selman. (1997). "Problem Structure in the Presence of Perturbations." In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, New Providence, RI, pp. 221–227. AAAI Press.

Gomes, C., B. Selman, and N. Crato. (1997). "Heavy-Tailed Distributions in Combinatorial Search." In *Proceedings of 6th International Conference on the Principles and Practice of Constraint Programming (CP-1997)*, Linz, Austria, Lecture Notes in Computer Science, Vol. 1330, pp. 121–135. New York: Springer.

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

Grotschel, M., L. Lovász, and A. Schrijver. (1993). *Geometric Algorithms and Combinatorial Optimization*. New York: Springer.

Guéret, C. and N. Jussien. (1999). "Combining AI/OR Techniques for Solving Open Shop Problems." In *Conference on Integration of Operations Research and Artificial Intelligence techniques in Constraint Programming (CP-AI-OR)*, Ferrara, Italy.

Heipcke, S. (1998). "Integrating Constraint Programming Techniques into Mathematical Programming." In *Proceedings of 13th European Conference on Artificial Intelligence*, Brigthon, UK, pp. 259–260. New York: Wiley.

Hill, B. (1975). "A Simple General Approach to Inference About the Tail of a Distribution." *Annals of Statistics* 3, 1163–1174.

Hogg, T. and B.C.W. Huberman (eds.). (1996). *Artificial Intelligence* 81(1–2), Special Issue on Phase Transitions and Complexity.

Hooker, J. (1996). "Resolution and the Integrality of Satisfiability Problems." *Mathematical Programming* 74, 1–10.

Hooker, J., G. Ottosson, E. Thorsteinsson, and H.-J. Kim. (1999). "On Integrating Constraint Propagation and Linear Programmimg for Combinatorial Optimization." In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, Orlando, FL, pp. 136–141. AAAI Press.

Ilog. (2001a). *Ilog cplex 7.1. User's Manual*.

Ilog. (2001b). *Ilog Solver 5.1. User's Manual*.

Jacobson, M. and P. Matthews. (1996). "Generating Uniformly Distributed Random Latin Squares." *Journal of Combinatorial Designs* 4(6), 405–437.

Kamath, A., N. Karmarkar, K. Ramakrishnan, and M. Resende. (1992). "A Continuous Approach to Inductive Inference." *Mathematical Programming* 57, 215–238.

Kautz, H., E. Horvitz, Y. Ruan, C. Gomes, and B. Selman. (2002). "Dynamic Restart Policies." In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada, pp. 674–671. AAAI Press.

Kirkpatrick, S. and B. Selman. (1994). "Critical Behavior in the Satisfiability of Random Boolean Expressions." *Science* 264, 1297–1301.

Kumar, S., A. Russell, and R. Sundaram. (1997). "Faster Algorithms for Optimal Switch Configuration." In *IEEE International Conference on Communications*.

Kumar, S., A. Russell, and R. Sundaram. (1999). "Approximating Latin Square Extensions." *Algorithmica* 24, 128–138.

Lawler, E. and J. Labetoulle. (1978). "On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming." *Journal of the Association for Computing Machinery* 25, 612–619.

Laywine, C. and G. Mullen. (1998). *Discrete Mathematics Using Latin Squares*, Wiley-Interscience Series in Discrete Mathematics and Optimization. New York: Wiley.

Little, J. and K. Darby-Dowman. (1995). "The Significance of Constraint Logic Programming to Operational Research." Operational Research Society, Invited Tutorial Papers.

Moskewicz, M., C. Madigan, Y. Zhao, L. Zhang, and S. Malik. (2001). "Chaff: Engineering and Efficient Sat Solver." In *Proceedings of the 38th Design Automation Conference (DAC 2000)*, Las Vegas, NV, pp. 530–535.

Motwani, R., J. Naor, and P. Raghavan. (1997). "Randomized Approximation Algorithms in Combinatorial Optimization." In D.S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems*. Boston, MA: PWS Publishing Company.

Plotkin, S., D. Shmoys, and E. Tardos. (1995). "Approximation Algorithms for Fractional Packing and Covering Problems." *Mathematics Operations Research* 20, 257–301.

Refalo, P. (1999). "Tight Cooperation and Its Application in Piecewise Linear Optimization." In *Proceedings of 5th International Conference on the Principles and Practice of Constraint Programming (CP-1999)*, Alexandria, VA, Lecture Notes in Computer Science, Vol. 1713, pp. 375–389. New York: Springer.

Refalo, P. (2000). “Linear Formulation of Constraint Programming Models and Hybrid Solvers.” In *Proceedings of 6th International Conference on the Principles and Practice of Constraint Programming (CP-2000)*, Singapore, Lecture Notes in Computer Science, Vol. 1894, pp. 369–383. New York: Springer.

Regin, J.-C. (1994). “A Filtering Algorithm for Constraints of Difference in CSP.” In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, pp. 362–367. AAAI Press.

Rodosek, R., M. Wallace, and M. Hajian. (1999). “A New Approach to Integrating Mixed Integer Programming with Constraint Logic Programming.” *Annals of Operations Research* 86, 63–87.

Ruan, Y., E. Horvitz, and H. Kautz. (2002). “Restart Policies with Dependence Among Runs: A Dynamic Programming Approach.” In *Proceedings of 8th International Conference on the Principles and Practice of Constraint Programming (CP-2002)*, Ithaca, NY, Lecture Notes in Computer Science, Vol. 2470, pp. 573–586. New York: Springer.

Shaw, P., K. Stergiou, and T. Walsh. (1998). “Arc Consistency and Quasigroup Completion.” In *Proceedings of Workshop on Binary Constraints, 13th European Conference on Artificial Intelligence*, Brigthon, UK. New York: Wiley.

Slaney, J., M. Fujita, and M. Stickel. (1995). “Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems.” *Computers and Mathematics with Applications* 29, 115–132.

von Neumann, J. (1953). “A Certain Zero-Sum Two-Person Game Equivalent to the Optimal Assignment Problem.” In *Contributions to the Theory of Games*, Vol. 2. Princeton: Princeton Univ. Press.

Walsh, T. (1999). “Search in a Small World.” In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99)*, Stockholm, Sweden, pp. 1172–1177.

Warners, J. (1999). “Nonlinear Approaches to Satisfiability Problems.” Ph.D. thesis, Technische Universiteit Eindhoven.