

# The Promise of LP to Boost CSP Techniques for Combinatorial Problems

---

**Carla P. Gomes**

Dept. of Comp. Science  
Cornell University  
Ithaca, NY 14853, USA  
email: [gomes@cs.cornell.edu](mailto:gomes@cs.cornell.edu)

**David B. Shmoys**

Dept. of Comp. Science  
School of Operations Research and Industrial Engineering  
Cornell University  
Ithaca, NY 14853, USA  
email: [shmoys@cs.cornell.edu](mailto:shmoys@cs.cornell.edu)

## **Abstract**

In recent years we have seen an increasing interest in combining Constraint Satisfaction Problem (CSP) methods and Linear Programming (LP) 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.

We propose a complete randomized backtrack search method for combinatorial problems that tightly couples CSP propagation techniques with randomized LP rounding. Our approach draws on recent results on approximation algorithms with theoretical guarantees, based on LP relaxations and randomized rounding techniques, as well on results that provide evidence that the run time distributions of combinatorial search methods are often heavy-tailed. 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., [8]). 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. Nevertheless, in a *purely combinatorial* setting, 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 (see e.g., [11, 12, 16, 23]), 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. Of course, also the cost of solving the LP relaxation should not outweigh the benefits in the reduction of search cost.

We propose a *complete* randomized backtrack search method that tightly couples CSP propagation techniques with randomized LP rounding. Our approach 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., [4, 19]), as well on results that uncovered the extreme variance or “unpredictability” in the run time of complete search procedures, often explained by the phenomenon of heavy-tailed cost distributions [10].

We use as a benchmark domain 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 [6]. 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 [18, 17].

We present our preliminary experimental findings for our randomized hybrid CSP/LP backtrack search method on 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 does improve over the pure strategies. In our hybrid approach, the LP relaxation with rounding strategy provides global information about the values to assign to the CSP variables. In effect, the randomized LP rounding provides powerful heuristic guidance to the CSP search, at least 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. 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 conjecture that the role of the LP relaxation in detecting infeasibility for pure combinatorial problems is not as important as its role as search heuristic. 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 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. A detailed empirical and theoretical evaluation is currently under way. 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 run time 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.

## 2 The Quasigroup Completion Problem

A quasigroup is an ordered pair  $(Q, \cdot)$ , where  $Q$  is a set of  $n$  symbols and  $(\cdot)$  is a binary operation on  $Q$  such that the equations  $a \cdot x = b$  and  $y \cdot a = b$  are uniquely solvable for every pair of elements  $a, b$  in  $Q$ . The *order*  $n$  of the quasigroup is the cardinality of the set  $Q$ .

The best way to understand the structure of a quasigroup is to consider its  $n$  by  $n$  multiplication table, as defined by its binary operation: The constraints of a quasigroup are such that its multiplication table defines a *Latin Square*. A Latin Square is an  $n \times n$  matrix on  $n$  symbols, such that each row/column is a permutation of its  $n$  symbols [18]. A *partial latin square PLS* is a partially filled  $n$  by  $n$  matrix such that no symbol occurs twice 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)<sup>1</sup> 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):

	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.

---

<sup>1</sup>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.

QCP is an NP-complete problem [6]. We have identified a phase transition phenomenon for the completion problem [9]. At the phase transition, problem instances switch from being almost all solvable (“under-constrained”) to being almost all unsolvable (“over-constrained”). The computationally hardest instances lie at the phase transition boundary. Figure 2 shows the median computational cost and phase transition. Along the horizontal axis we vary the ratio of pre-assigned cells.<sup>2</sup> We note that even though all the instances are from an NP-complete problem, we clearly distinguish various regions of problem difficulty. In particular, both at low ratios and high ratios of preassigned colors the median solution cost is relatively small. However, in between these two regimes, the complexity peaks and, in fact, exhibits strong exponential growth.<sup>3</sup>

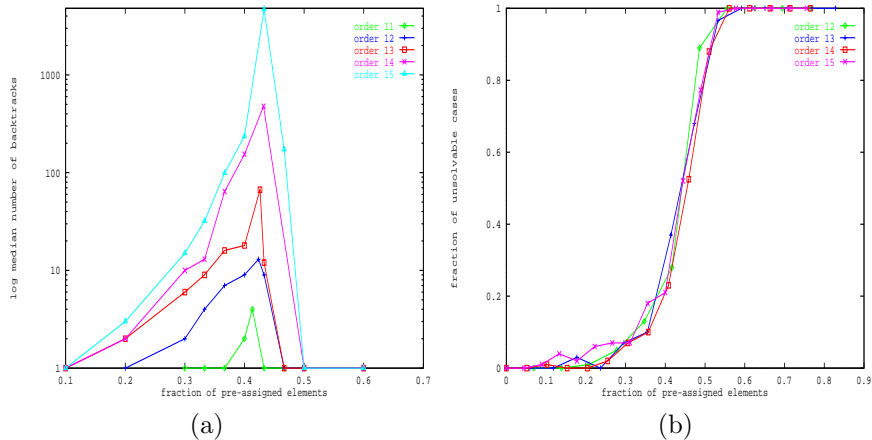


Figure 2: (a) Cost profile, and (b) phase transition for the quasigroup completion problem (up to order 15).

We identified the erratic behavior of the mean and the variance of different randomized backtrack style procedures running on instances of QCP [10]. Figure 3 (a) depicts this phenomenon, displaying the mean cost 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 instance is easy, the median number of backtracks for solution is 1, some runs take more than  $10^6$  backtracks. Figure 3 (b) plots the log-log plot of the tail (*i.e.*,  $(1-F(x))$ ) of the runtime distributions of a randomized backtrack search method on three QCP instances: one instance in the under-constrained area (the same instance of order 11 with 64% of holes), one in the critically constrained area, and one in the medium constrained area. The linear nature of the long tails in this log-log plot directly reveals the phenomenon of heavy-tails.

The formal explanation for heavy-tailed behavior comes from the fact that there is a

<sup>2</sup>Note that the ratio of pre-assigned cells corresponds to the complement of the ratio of holes, *i.e.*,  $1 - h/n^2$ .

<sup>3</sup>The exact location of the phase transition appears to be characterized in terms of  $(1-h)/n^p$ , where  $p$  is around 1.55. However, and given that for low orders of quasigroups  $p = 2$  is a good approximation, for simplification we talk about proportion of preassigned colors in terms of the total number of cells of the matrix, *i.e.*,  $N^2$  [1].

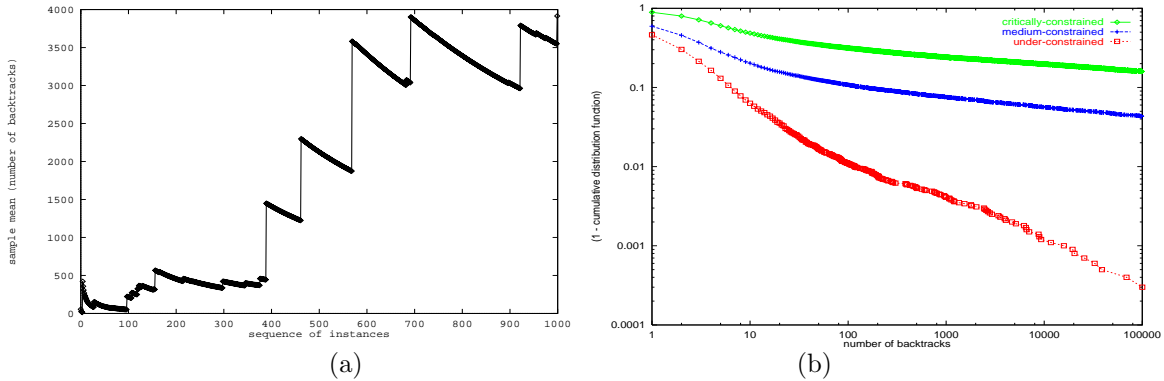


Figure 3: (a) Erratic behavior of mean cost value. (b) Log-log plot of heavy-tailed behavior.

non zero probability of entering a subtree of exponential size that has no solutions [3]. 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 [10].

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 [17]. 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.

### 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 [9]:

$$\begin{aligned}
 & x_{i,j} \in \{1, \dots, n\} \quad \forall i, j \\
 & x_{i,j} = k \quad \forall i, j \text{ such that } PLS_{ij} = k \\
 & \text{alldiff } (x_{i,1}, x_{i,2}, \dots, x_{i,n}) \quad \forall i = 1, 2, \dots, n \\
 & \text{alldiff } (x_{1,j}, x_{2,j}, \dots, x_{n,j}) \quad \forall j = 1, 2, \dots, n
 \end{aligned}$$

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 around 33 relatively well [9, 22, 1]. However, for higher orders, instances in the critically constrained area are beyond the reach of pure CSP solvers, given the highly exponential behavior in this region.

### 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 [17]:

$$\begin{aligned} & \max \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n x_{i,j,k} \\ & \text{subject to} \\ & \sum_{i=1}^n x_{i,j,k} \leq 1, \quad \forall j, k \\ & \sum_{j=1}^n x_{i,j,k} \leq 1, \quad \forall i, k \\ & \sum_{k=1}^n x_{i,j,k} \leq 1, \quad \forall i, j \\ & x_{i,j,k} = 1 \quad \forall i, j, k \text{ such that } PLS_{ij} = k \\ & x_{i,j,k} \in \{0, 1\} \quad \forall i, j, k \\ & i, j, k = 1, 2, \dots, n \end{aligned}$$

If PLS is completable, the optimal value of this integer program is  $h$ , *i.e.*, the number of holes in  $PLS$ . Kumar et al. [17] 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  $x_{i,j,k} \geq 0$ . 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 et al. 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 et al. solve problems up to order 9.

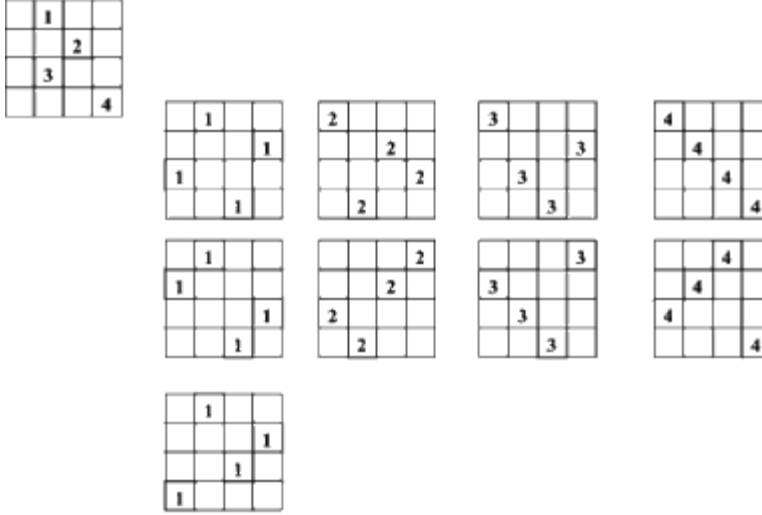


Figure 4: Families of compatible matchings for the partial latin square in the left upper corner. For example, the family of compatible matchings for symbol 1 has three compatible matchings.

### 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  $k$  so that there is exactly one cell colored  $k$  in every row and column. We shall call one such collection of cells a *compatible matching* for  $k$ . 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.

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

$$\sum_{M \in \mathcal{M}_k} y_{k,M} = 1, \quad \forall k$$

$$\sum_{k=1}^n \sum_{M \in \mathcal{M}_k: (i,j) \in M} y_{k,M} \leq 1, \quad \forall i, j$$

$$y_{k,M} \in \{0, 1\} \quad \forall k, M.$$

Once again, we can consider the linear programming relaxation of this formulation, in which the binary constraints are relaxed to be merely non-negativity constraints. It is significant to note that, for any feasible solution  $y$  to this linear programming relaxation, one can generate a corresponding feasible solution  $x$  to the assignment formulation, by simply computing  $x_{i,j,k} = \sum_{M \in \mathcal{M}_k} y_{k,M}$ . 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  $n$ . In spite of this difficulty, one may apply column generation techniques (see, e.g., the textbook by [5]) 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 in which good solutions are computed for an integer programming problem in which the variables are 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.

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 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.

This approach has led to striking results in a number of settings. For example, Goemans and Williamson [7] 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  $k$ . 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 [21]. 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. In fact, 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 it is straightforward to show that the expected number of cells for which some such conflict exists is at most  $h/e$ ; that is, at least  $(1 - 1/e)h$  holes can be expected to be filled by this technique.

## 5 Hybrid CSP/LP Randomized Rounding Backtrack Search

We now describe a complete randomized backtrack search algorithm for the quasigroup (latin square) completion problem.

A central feature of the algorithm is the fact that it maintains two different formulations of the quasigroup completion problem: the CSP formulation, as described in section 3.1, and the relaxation of the LP formulation described in section 3.2.<sup>4</sup> The hybrid nature of the algorithm results from the combination of strategies for variable and value assignment, and propagation, based on the two underlying models.

The algorithm is initialized by populating the CSP model and propagating constraints over this model. The CSP model is implemented in Ilog/Solver [14]. For the propagation of the ALLDIFF constraint we use the extended version provided by Ilog [14, 20]. The updated domain values are then used to populate the LP model. We solve the LP model using Ilog/Cplex Barrier [13].

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.

At the top of the backtrack search tree, variable and value selection are based on the LP rounding. After each value assignment based on the LP, full propagation is performed on the CSP model. The percentage of variables set by the LP is controlled by the parameter %LP. (So, with %LP = 0, we have a pure CSP strategy.) 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

---

<sup>4</sup>In the experiments reported here, we are using the assignment formulation for the LP. Even though the packing formulation has stronger theoretical bounds, because we solve this formulation using column generation, it is more difficult to show a concrete payoff of this approach, in practice. We are pursuing further experiments with this approach.

variable settings. However, we have found that this information can be computed more efficiently through the CSP model.

Ideally, in order to increase the accuracy of the variable assignments based on LP-rounding, 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 our LP rounding strategy, 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. In the search guided by the CSP model, we use a variant of the Brelaz heuristic [2, 9] for the variable and value selections.

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 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 with only a few additional backtracks can uncover the same information.

In this setting, we are effectively using the LP values as heuristic guidance, using a randomized rounding approach inspired by the rounding schemes used in approximation algorithms. As we discussed in section 3.3, for the packing formulation of the LP, we have a clear theoretical basis for such a rounding scheme. For the assignment formulation, the theoretical justification is less immediate — nevertheless, as we will see below, our rounding scheme leads to a clear practical payoff.

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. In the limit, we run the algorithm without a cutoff.

## 6 Empirical Results

To investigate our hypothesis that the LP relaxation 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

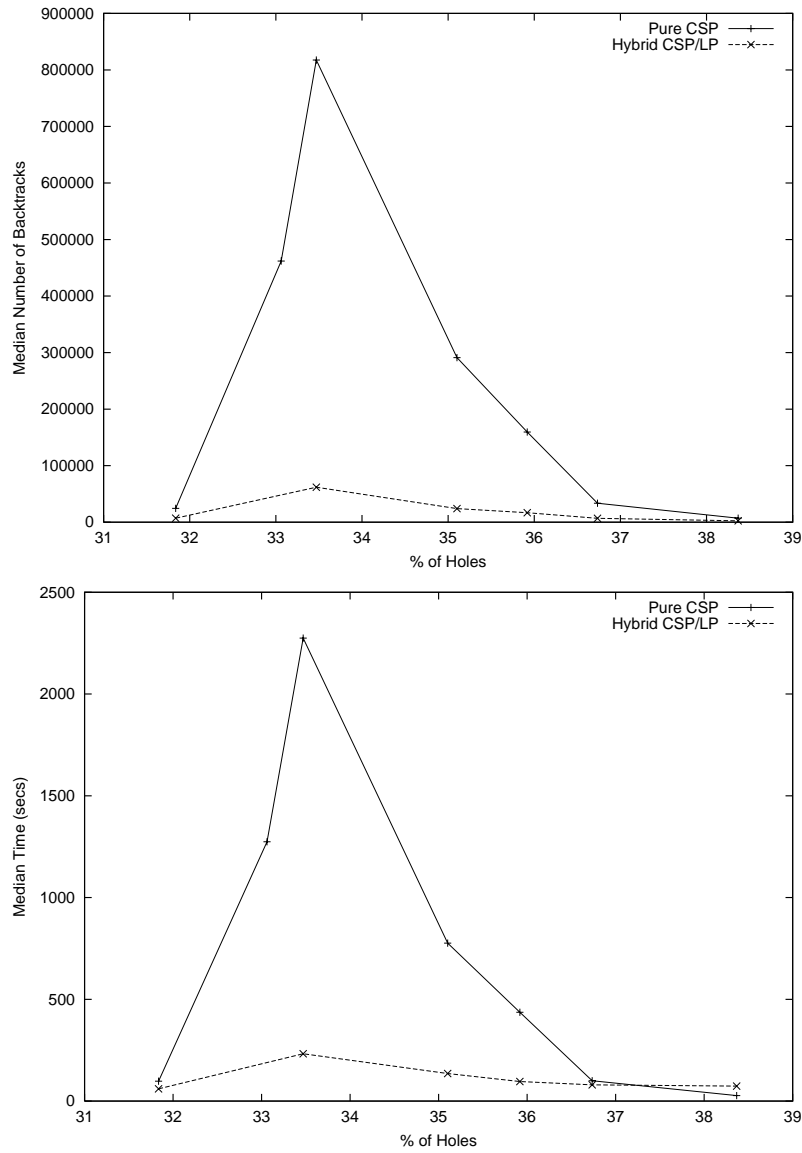


Figure 5: Median run time (secs) for QWH instances of order 35. (100 instances per data point)

latin square is generated using a Markov chain Monte Carlo shuffling process [15]. 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.<sup>5</sup> We can again finely tune the complexity of the completion task by varying the fraction of the uncolored cells [1].

<sup>5</sup>The code for this generator is available by contacting Carla Gomes (gomes@cs.cornell.edu).

In Figure 5, 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 run time (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 run time. The relative payoff of our hybrid strategy is largest for the hardest problem instances (around 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 — 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  $\%LP \geq 10$ , we solve the instance on each run. Looking at the overall run time as a function of  $\%LP$ , we see that at some point further use of LP relaxations becomes counterproductive. The best performance is obtained with  $\%LP$  around 20.

$\% LP$	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 1: Hybrid CSP/LP search on an instance of order 35 with 405 holes.

In Table 2 and Table 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. The success rate increases with a higher  $\%LP$ . From the median overall run time, we see that the best performance is obtained for  $\%LP$  around 25.

We have not found any instance that could be solved with the pure CSP strategy and could not be solved with the CSP/LP strategy. Furthermore, hard instances of QCP/QWH appear 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.

% 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

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%	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

Table 3: Instance of order 40, with 544 holes.

Overall, our hybrid method significantly extends the range of QWH problems we can solve. The hybrid strategy allows us to improve on the time performance of the pure CSP strategy and reliably solve larger instances, up to order 40 to 45. We also solved several hard instances of order 50. On our very hardest problem, we had to increase %LP to around 50%. So, apparently, more guidance was required from the LP relaxation.

## 7 Conclusions

Constraint based search techniques have shown to be remarkably efficient on purely combinatorial problems. In many domains, such techniques outperform integer programming based approaches. Nevertheless, LP relaxations may still provide useful information, for example, in guiding constraint based search techniques.

We have demonstrated the promise of boosting CSP methods using LP relaxations on hard, purely combinatorial problems. Our approach involves a randomized rounding strategy inspired by recent rounding methods used in approximation algorithms. In this setting, the LP 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 between the CSP and LP method. We

simultaneously maintain a CSP and an LP model of the problem. 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 rounding decisions stay accurate during the search process. The continuous interleaving of CSP propagation and LP heuristic guidance using randomized rounding 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 half 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 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 and LP formulations and different rounding strategies. In particular, we are currently experimenting with the packing formulation which offers better theoretical guarantees.

## References

- [1] D. Achlioptas, Carla Gomes, Henry Kautz, and Bart Selman. Generating Satisfiable Instances. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, New Providence, RI, 2000. AAAI Press.
- [2] D. Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
- [3] H. Chen, C. Gomes, and B. Selman. Formal models of heavy-tailed behavior in combinatorial search. In *Proceedings of 7th Intl. Conference on the Principles and Practice of Constraint Programming (CP-2001)*, *Lecture Notes in Computer Science, Vol. 2239*, Springer-Verlag, pages 408–422, 2001.
- [4] F. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. In *Submitted for publication*, 1999. Preliminary version of this paper (with the same title) appeared in proceedings of the Sixth Conference on Integer Programming and Combinatorial Optimization.
- [5] Vasek Chvatal. *Linear Programming*. W.H.Freeman Company, 1983.
- [6] C. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, (8):25–30, 1984.
- [7] M. X. Goemans and D. P. Williamson. 0.878-approximation algorithms for max-cut and max-sat. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Theory of Computing*, pages 422–431, 1994.
- [8] Carla Gomes. editor, *The Knowledge Engineering Review. Special Issue on the Integration of Artificial Intelligence and Operations Research Techniques, Vol. 15 (1) and Vol. 16 (1)*. Cambridge Press, 2000-2001.

- [9] Carla Gomes and Bart Selman. Problem Structure in the Presence of Perturbations. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 221–227, New Providence, RI, 1997. AAAI Press.
- [10] Carla Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. of Automated Reasoning*, 24(1-2):67–100, 2000.
- [11] John Hooker. Resolution vs. cutting plane solution of inference problems: Some computational experience. *Operations Research Letter*, 7(1):1–7, 1988.
- [12] John Hooker. Resolution and the integrality of satisfiability problems. *Mathematical Programming*, 74:1–10, 1996.
- [13] Ilog Inc. Ilog cplex 7.1. user’s manual., 2001.
- [14] Ilog Inc. Ilog solver 5.1. user’s manual., 2001.
- [15] M.T. Jacobson and P. Matthews. Generating uniformly distributed random latin squares. *J. of Combinatorial Designs*, 4(6):405–437, 1996.
- [16] A.P. Kamath, N. K. Karmarkar, K. G. Ramakrishnan, and M. G. C. Resende. A continuous approach to inductive inference. *Mathematical Programming*, 57:215–238, 1992.
- [17] S. R. Kumar, A. Russell, and R. Sundaram. Approximating latin square extensions. *Algorithmica*, 24:128–138, 1999.
- [18] Charles Laywine and Gary Mullen. *Discrete Mathematics using Latin Squares*. Wiley-Interscience Series in Discrete mathematics and Optimization, 1998.
- [19] Rajeev Motwani, Joseph Naor, and Prabhakar Raghavan. Randomized approximation algorithms in combinatorial optimization. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.
- [20] J. C. Regin. A filtering algorithm for constraints of difference in csp. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 362–367, Seattle, WA, 1994. AAAI Press.
- [21] R. Regis, C. Gomes, and D. Shmoys. An improvement performance guarantee for the partial latin square problem. Manuscript in preparation, 2002.
- [22] P. Shaw, K. Stergiou, and T. Walsh. Arc consistency and quasigroup completion. In *Proceedings of ECAI-98, workshop on binary constraints*, 1998.
- [23] Johannes Warners. *Nonlinear approaches to satisfiability problems*. PhD thesis, Technische Universiteit Eindhoven, 1999.