# Counting Solutions of Knapsack Constraints

Gilles Pesant and Claude-Guy Quimper

Département de génie informatique et génie logiciel
École Polytechnique de Montréal
C.P. 6079, succ. Centre-ville
Montreal, Canada H3C 3A7
pesant@crt.umontreal.ca, claude-guy.quimper@polymtl.ca

**Abstract.** This paper furthers the recent investigation of search heuristics based on solution counting information, by proposing and evaluating algorithms to compute solution densities of variable-value pairs in knapsack constraints. Given a domain consistent constraint, our first algorithm is inspired from what was proposed for regular language membership constraints. Given a bounds consistent constraint, our second algorithm draws from discrete uniform distributions. Experiments on several problems reveal that simple search heuristics built from the information obtained by these algorithms can be very effective.

## 1 Introduction

Recent work on search heuristics using information about the number of solutions of constraints has shown encouraging results to solve constraint satisfaction problems [4, 8]. Working at the level of individual constraints, it asks not only whether there exists a solution in which variable $x$ takes value $d$, which corresponds to the familiar concept of consistency, but also *how many* of the solutions feature that particular assignment. Such an approach requires efficient ways to answer that question for each type of constraint commonly found in constraint programs. This paper examines knapsack constraints, present in many problems.

The $\texttt{knapsack}(\mathbf{x}, \mathbf{c}, \ell, u)$ constraint holds if

$$\ell \leq \mathbf{cx} \leq u$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is an integer row vector, $\mathbf{x}$ is a column vector of finite domain variables $(x_1, x_2, \ldots, x_n)^T$ with $x_i \in D_i$, and $\ell$ and $u$ are integers. To be interpreted as a knapsack, the integer values involved (including those in the finite domains) are non negative. We will come back to this restriction in Section 5. We assume that $l$ and $u$ are finite as they can always be set to the smallest and largest value that $\mathbf{cx}$ can take.

To prepare us to manipulate information on the number of solutions of knapsack constraints, we recall some definitions and notation from [4, 8].

**Definition 1 (solution count).** *Given a constraint $\gamma(x_1, \ldots, x_k)$ and respective finite domains $D_i$ $_{1 \leq i \leq k}$, let $\#\gamma(x_1, \ldots, x_k)$ denote the number of solutions of constraint $\gamma$, called its* solution count.

**Definition 2 (solution density).** *Given a constraint $\gamma(x_1, \ldots, x_k)$, respective finite domains $D_i$ $_{1 \leq i \leq k}$, a variable $x_i$ in the scope of $\gamma$, and a value $d \in D_i$, we will call*

$$\sigma(x_i, d, \gamma) = \frac{\#\gamma(x_1, \ldots, x_{i-1}, d, x_{i+1}, \ldots, x_k)}{\#\gamma(x_1, \ldots, x_k)}$$

*the* solution density *of pair $(x_i, d)$ in $\gamma$. It measures how often a certain assignment is part of a solution.*

In the rest of the paper, Section 2 presents the counting algorithm for knapsack constraints on which domain consistency is enforced, Section 3 presents another counting algorithm for bounds consistent knapsack constraints, and Section 4 reports several experiments. Final comments are given in Section 5.

## 2 Counting with Domain Consistent Knapsacks

In [6], Trick proposes a filtering algorithm for knapsack constraints that relies on a graph whose structure is very similar to that of the `regular` constraint [3]. He notes that every path in that graph corresponds to a solution and that counting the number of solutions is easily obtained through a recursion without the need to enumerate these solutions. Not surprisingly then, the computation of solution counts and solution densities for knapsack constraints follows quite directly from the work on the `regular` constraint.

We start from the reduced graph described in [6], which is a layered directed graph $G(V, A)$ with a vertex $v_{i,b} \in V$ for $1 \leq i \leq n$ and $0 \leq b \leq u$ whenever

$$\forall\, j \in [1, i],\ \exists\, d_j \in D_j \text{ such that } \sum_{j=1}^{i} c_j d_j = b$$

and

$$\forall\, j \in (i, n],\ \exists\, d_j \in D_j \text{ such that } \ell - b \leq \sum_{j=i+1}^{n} c_j d_j \leq u - b,$$

and an arc $(v_{i,b}, v_{i+1,b'}) \in A$ whenever

$$\exists\, d \in D_{i+1} \text{ such that } c_{i+1} d = b' - b.$$

We define the following two recursions to represent the number of incoming and outgoing paths at each node.

For every vertex $v_{i,b} \in V$, let $\#ip(i, b)$ denote the number of paths from vertex $v_{0,0}$ to $v_{i,b}$:

$$\begin{aligned} \#ip(0, 0) &= 1 \\ \#ip(i+1, b') &= \sum_{(v_{i,b}, v_{i+1,b'}) \in A} \#ip(i, b), \quad 0 \leq i < n \end{aligned}$$

Let $\#op(i,b)$ denote the number of paths from vertex $v_{i,b}$ to a vertex $v_{n,b'}$ with $\ell \leq b' \leq u$.

$$\begin{aligned} \#op(n,b) &= 1 \\ \#op(i,b) &= \sum_{(v_{i,b},v_{i+1,b'})\in A} \#op(i+1,b'), \quad 0 \leq i < n \end{aligned}$$

The total number of paths (i.e. the *solution count*) is given by

$$\#\texttt{knapsack}(\mathbf{x},\mathbf{c},\ell,u) = \#op(0,0)$$

in time linear in the size of the graph even though there may be exponentially many of them. The *solution density* of a variable-value pair $(x_i,d)$ is given by

$$\sigma(x_i,d,\texttt{knapsack}) = \frac{\sum_{(v_{i-1,b},v_{i,b+c_i d})\in A} \#ip(i-1,b)\cdot\#op(i,b+c_i d)}{\#op(0,0)}.$$

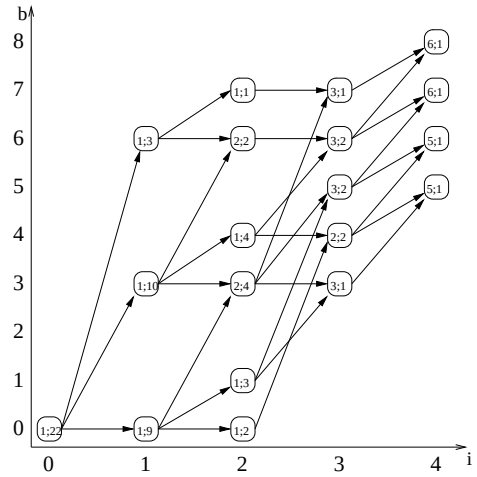


**Fig. 1.** Reduced graph for knapsack constraint $5 \leq 3x_1 + x_2 + 2x_3 + x_4 \leq 8$ with $D_1 = \{0,1,2\}, D_2 = \{0,1,3\}, D_3 = \{0,1,2\}, D_4 = \{1,2\}$. Vertex labels represent the number of incoming and outgoing paths.

In Figure 1, the left and right labels inside each vertex give the number of incoming and outgoing paths for that vertex, respectively. Table 1 reports the solution densities for every variable-value pair.

The time required to compute these recursions is related to the number of arcs, which is in $\mathcal{O}(nu \max_{1\leq i\leq n}\{|D_i|\})$. Then each solution density computes a summation over a subset of the arcs but each arc of the graph is involved in at most one such summation, so the overall time complexity of computing every solution density is $\mathcal{O}(nu \max_{1\leq i\leq n}\{|D_i|\})$ as well.

| | variable | | | |
|---|---|---|---|---|
| value | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
| 0 | 9/22 | 8/22 | 9/22 | – |
| 1 | 10/22 | 8/22 | 7/22 | 11/22 |
| 2 | 3/22 | – | 6/22 | 11/22 |
| 3 | – | 6/22 | – | – |

**Table 1.** Solution densities for the example of Fig. 1.

## 3 Counting with Bounds Consistent Knapsacks

Knapsack constraints, indeed most arithmetic constraints, have traditionally been handled by enforcing bounds consistency, a much cheaper form of inference. In some situations, we may not afford to enforce domain consistency in order to get the solution counting information we need to guide our search heuristic. Can we still retrieve such information, perhaps not as accurately, from the weaker bounds consistency?

Consider the variable $x$ with domain $D = [a, b]$. Each value in $D$ is equiprobable. We associate to $x$ the discrete random variable $X$ which follows a discrete uniform distribution with probability mass function $f(v)$, mean $\mu = E[X]$, and variance $\sigma^2 = Var[X]$.

$$f(v) = \begin{cases} \frac{1}{b-a+1} \text{ if } a \leq v \leq b \\ 0 \quad \text{otherwise} \end{cases} \tag{1}$$

$$\mu = \frac{a+b}{2} \tag{2}$$

$$\sigma^2 = \frac{(b-a+1)^2 - 1}{12} \tag{3}$$

To find the distribution of a variable subject to a knapsack constraint, one needs to find the distribution of a linear combination of uniformly distributed random variables. Lyapunov's central limit theorem allows us to approximate the distribution of such a linear combination.

**Theorem 1 (Lyapunov's central limit theorem).** *Consider the independent random variables $X_1, \ldots, X_n$. Let $\mu_i$ be the mean of $X_i$, $\sigma_i^2$ be its variance, and $r_i^3 = E[|X_i - \mu_i|^3]$ be its third central moment. If*

$$\lim_{n\to\infty} \frac{(\sum_{i=1}^n r_i^3)^{\frac{1}{3}}}{(\sum_{i=1}^n \sigma_i^2)^{\frac{1}{2}}} = 0,$$

*then the random variable $S = \sum_{i=1}^n X_i$ follows a normal distribution with mean $\mu_S = \sum_{i=1}^n \mu_i$ and variance $\sigma_S^2 = \sum_{i=1}^n \sigma_i^2$.*

The probability mass function of the normal distribution with mean $\mu$ and variance $\sigma^2$ is the Gaussian function:

$$\varphi(x) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \tag{4}$$

Note that Lyapunov's central limit theorem does not assume that the variables are taken from identical distributions. This is necessary since variables with different domains have different distributions.

Lemma 1 defines an upper bound on the third central moment of the expression $kX$ where $k$ is a positive coefficient and $X$ is a uniformly distributed random variable.

**Lemma 1.** *Let $Y$ be a discrete random variable equal to $kX$ such that $k$ is a positive coefficient and $X$ is a discrete random variable uniformly distributed over the interval $[a, b]$. The third central moment $r^3 = E[|Y - E[Y]|^3]$ is no greater than $k^3(b-a)^3$.*

*Proof.* The case where $a = b$ is trivial. We prove for $b - a > 0$. The proof involves simple algebraic manipulations from the definition of the expectation.

$$r^3 = \sum_{i=ka}^{kb} |i - E[Y]|^3 f(i) \tag{5}$$

$$= \sum_{j=a}^{b} |kj - kE[X]|^3 f(j) \tag{6}$$

$$= k^3 \sum_{j=a}^{b} \left| j - \frac{a+b}{2} \right|^3 \frac{1}{b-a+1} \text{ since } k > 0 \tag{7}$$

$$= \frac{k^3}{b-a+1} \left( \sum_{j=a}^{\frac{a+b}{2}} \left( \frac{a+b}{2} - j \right)^3 + \sum_{j=\frac{a+b}{2}}^{b} \left( j - \frac{a+b}{2} \right)^3 \right) \tag{8}$$

$$= \frac{k^3}{b-a+1} \left( \sum_{j=0}^{\frac{b-a}{2}} j^3 + \sum_{j=0}^{\frac{b-a}{2}} j^3 \right) \tag{9}$$

$$\leq \frac{2k^3}{b-a} \sum_{j=0}^{\frac{b-a}{2}} j^3 \text{ since } b - a > 0 \tag{10}$$

Let $m = \frac{b-a}{2}$.

$$
\begin{aligned}
r^3 &\leq \frac{k^3}{m} \sum_{j=0}^{m} j^3 && (11)\\
&\leq \frac{k^3}{m} \left( \frac{1}{4}(m+1)^4 - \frac{1}{2}(m+1)^3 + \frac{1}{4}(m+1)^2 \right) && (12)\\
&\leq \frac{k^3}{m} \left( \frac{m^4}{4} + \frac{m^3}{2} + \frac{m^2}{4} \right) && (13)\\
&\leq \frac{k^3}{m} \left( \frac{m^4}{4} + m^4 + m^4 \right) \text{ since } m \geq \tfrac{1}{2} && (14)\\
&\leq \frac{9}{4} k^3 m^3 && (15)
\end{aligned}
$$

Which confirms that $r^3 \leq \frac{9}{32} k^3 (b-a)^3 \leq k^3 (b-a)^3$. □

Lemma 2 defines the distribution of a linear combination of uniformly distributed random variables.

**Lemma 2.** *Let $Y = \sum_{i=1}^{n} c_i X_i$ be a random variable where $X_i$ is a discrete random variable uniformly chosen from the interval $[a_i, b_i]$ and $c_i$ is a non-negative coefficient. When $n$ tends to infinity, the distribution of $Y$ tends to a normal distribution with mean $\sum_{i=1}^{n} c_i \frac{a_i+b_i}{2}$ and variance $\sum_{i=1}^{n} c_i^2 \frac{(b_i-a_i+1)^2-1}{12}$.*

*Proof.* Let $Y_i = c_i X_i$ be a random variable. We want to characterize the distribution of $\sum_{i=1}^{n} Y_i$. Let $m_i = \frac{b_i-a_i}{2}$. The variance of the uniform distribution over the interval $[a_i, b_i]$ is $\sigma_i^2 = \frac{(b_i-a_i+1)^2-1}{12} = \frac{(m_i+\frac{1}{2})^2}{3} - \frac{1}{12}$. We have $Var[Y_i] = c_i^2 Var[X_i] = c_i^2 \sigma_i^2$. Let $r_i^3$ be the third central moment of $Y_i$. By Lemma 1, we have $r_i^3 \leq c_i^3 (b_i - a_i)^3$. Let $L$ be the term mentioned in the condition of Lyapunov's central limit theorem:

$$
L = \lim_{n \to \infty} \frac{\left( \sum_{i=1}^{n} r_i^3 \right)^{\frac{1}{3}}}{\left( \sum_{i=1}^{n} c_i^2 \sigma_i^2 \right)^{\frac{1}{2}}} \tag{16}
$$

Note that the numerator and the denominator of the fraction are non-negative. This implies that $L$ itself is non-negative. We prove that $L \leq 0$ as $n$ tends to infinity.

$$L \leq \lim_{n\to\infty} \frac{\left(\sum_{i=1}^{n} 8c_i^3 m_i^3\right)^{\frac{1}{3}}}{\left(\sum_{i=1}^{n} c_i^2 \left(\frac{(m_i+\frac{1}{2})^2}{3} - \frac{1}{12}\right)\right)^{\frac{1}{2}}} \tag{17}$$

$$\leq \lim_{n\to\infty} \frac{\left(8\sum_{i=1}^{n} c_i^3 m_i^3\right)^{\frac{1}{3}}}{\left(\frac{1}{3}\sum_{i=1}^{n} c_i^2 m_i^2\right)^{\frac{1}{2}}} \tag{18}$$

$$\leq \lim_{n\to\infty} 2\sqrt{3}\sqrt[6]{\frac{\left(\sum_{i=1}^{n} c_i^3 m_i^3\right)^2}{\left(\sum_{i=1}^{n} c_i^2 m_i^2\right)^3}} \tag{19}$$

$$\leq \lim_{n\to\infty} 2\sqrt{3}\sqrt[6]{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(c_i c_j m_i m_j)^3}{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(c_i c_j c_k m_i m_j m_k)^2}} \tag{20}$$

Note that in the last inequality, the terms $(c_i c_j m_i m_j)^3$ and $(c_i c_j c_k m_i m_j m_k)^2$ are of the same order. However, there are $n$ times more terms in the denominator than the numerator. Therefore, when $n$ tends to infinity, the fraction tends to zero which proves that $L = 0$ as $n$ tends to zero.

By Lyapunov's central limit theorem, as $n$ tends to infinity, the expression $Y = \sum_{i=1}^{n} Y_i$ tends to a normal distribution with mean $E[Y] = \sum_{i=1}^{n} c_i E[X_i] = \sum_{i=1}^{n} c_i \frac{a_i+b_i}{2}$ and variance $Var[Y] = \sum_{i=1}^{n} c_i^2 Var[X_i] = \sum_{i=1}^{n} c_i^2 \frac{(b_i-a_i+1)^2-1}{12}$. □

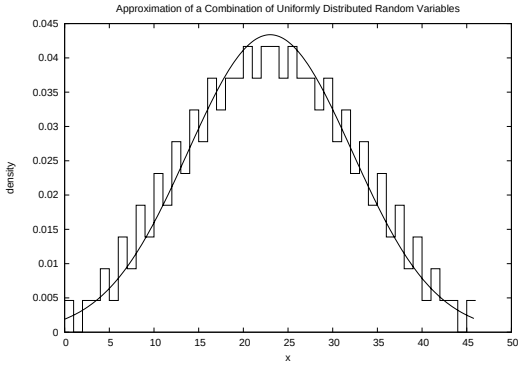


**Fig. 2.** The histogram is the actual distribution of the expression $3x + 4y + 2z$ for $x, y, z \in [0, 5]$. The curve is the approximation given by the Gaussian curve with mean $\mu = 22.5$ and variance $\sigma^2 = 84.583$.

Consider the knapsack constraint $\ell \leq \sum_{i=1}^{n} c_i x_i \leq u$. Let $x_{n+1}$ be a variable with domain $D_{n+1} = [\ell, u]$. We obtain $x_j = \frac{1}{c_j}(x_{n+1} - \sum_{i=1}^{j-1} c_i x_i - \sum_{i=j+1}^{n} c_i x_i)$.

Some coefficients in this expression might be negative. They can be made positive by setting $c'_i = -c_i$ and $D'_i = [-\max(D_i), -\min(D_i)]$. When $n$ grows to infinity, the distribution of $x_j$ tends to a normal distribution as stated in Lemma 2. In practice, the normal distribution is a good estimation even for small values of $n$. Figure 3 shows the actual distribution of the expression $3x + 4y + 2z$ for $x, y, z \in [0, 5]$ and its approximation by a normal distribution.

Given a variable $x_i$ subject to a knapsack constraint, Algorithm 1 returns the assignment $x_i = v$ with the highest solution density. The *for loop* computes the average mean $\mu_i$ and the variance $\sigma_i^2$ of the uniform distribution associated to each variable $x_i$. Lines 4 to 5 compute the mean and the variance of the distribution of $x_{n+1} - \sum_{j=1}^{n} c_j x_j$ while Lines 6 and 7 compute the mean and the variance of $x_i = \frac{1}{c_i}(x_{n+1} - \sum_{j=1}^{i-1} c_j x_j - \sum_{j=i+1}^{n} c_j x_j)$. Since this normal distribution is symmetric and unimodal, the most likely value $k_i$ in the domain $D_i$ is the one closest to the mean $\mu_i$. The algorithm finds and returns this value as well as its density $d_i$. The density $d_i$ is computed using the normal distribution. Since the variable $x_i$ must be assigned to a value in its domain, the algorithm normalizes on Line 9 the distribution over the values in the interval $[\min(D_i), \max(D_i)]$.

```
1  for j ∈ [1, n] do
2      μ_j ← (min(D_j)+max(D_j))/2;
3      σ_j^2 ← ((max(D_j)−min(D_j)+1)^2−1)/12;
4  M ← (l+u)/2 − Σ_{j=1}^{n} c_j μ_j;
5  V ← ((u−l+1)^2−1)/12 + Σ_{j=1}^{n} c_j^2 σ_j^2;
6  m ← (M+c_i μ_i)/c_i;
7  v ← (V−c_i^2 σ_i^2)/c_i^2;
8  k_i ← argmin_{k∈D_i} |k − m|;
9  d_i ← e^{−(k_i−m)^2/2v} / Σ_{k=min(D_i)}^{max(D_i)} e^{−(k−m)^2/2v};
10 return ⟨x_i = k_i, d_i⟩
```

**Algorithm 1**: FindDensity($[X_1, \ldots X_n]$, $i$) returns the assignment $x_i = k$ with the highest density as well as its density $d_i$.

Lines 1 through 5 take $O(n)$ time to execute. Line 8 depends on the data structure used by the solver to encode a domain. We assume that the line takes $O(\log |D_i|)$ time to execute. The summation on Line 9 can be computed in constant time by approximating the summation with $\Phi_{m,v}(\max(D_i) + \frac{1}{2}) - \Phi_{m,v}(\min(D_i) + \frac{1}{2})$ where $\Phi_{m,v}$ is the normal cumulative distribution function with average $m$ and variance $v$. The constant $\frac{1}{2}$ is added for the continuity correction. Other lines have a constant running time. The total complexity of Algorithm 1 is therefore $O(n+\log |D_i|)$. Note that Line 1 to Line 5 do not depend on the value of $i$. Their computation can therefore be cached for subsequent calls

to the function over the same knapsack constraint. Using this technique, finding the variable $x_i \in \{x_1, \ldots, x_n\}$ which has an assignment $x_i = k$ of maximum density takes $O(\sum_{i=1}^{n} \log |D_i|)$ time.

A source of alteration of the distribution are values in the interval which are absent from the actual domain. Bounds consistency approximates the domain of a variable with its smallest covering interval. In order to reduce the error introduced by this approximation, one can compute the actual mean and actual variance of a domain $D_i$ on Lines 2 and 3 instead of using the mean and the variance of the covering interval, at a revised overall cost of $O(\sum_{i=1}^{n} |D_i|)$.

## 4 Experiments

We evaluated the usefulness of solution counting information from knapsack constraints on four types of problems. The first two are benchmarks from the literature and feature 0-1 knapsack constraints. The third one is a magic square completion problem, featuring integer knapsack constraints. The last one is inspired from the area of rostering and features integer knapsack constraints as well. These problems were chosen because they are modeled using (almost) only knapsack constraints, in order to avoid the separate issue of combining heuristic information from different types of constraints.

A word on the notation used for the search heuristics investigated. The prefix "`Lexico`" refers to variable selection according to lexicographic order, whereas "`Dom`" refers to variable selection in increasing order of domain size. Keyword "`Random`" refers to random value selection. Keyword "`MaxSD`" refers to variable/value selection in decreasing order of solution density, or solely as a value selection heuristic if a prefix indicates a particular variable selection heuristic. Note that densities are considered separately from every constraint. In the case of bounds consistency, keyword "`MaxSD+`" indicates that the actual mean and variance were computed from the domains in Algorithm 1.

The experiments were performed with ILOG Solver 5.1 on a Sun Fire 4800 (1.2 GHz CPU, 16 Gb RAM, 43 Gb swap) running under SunOS 5.10. Measures reported for heuristics involving random choices are an average over five runs.

### 4.1 Market Split Problem

The Market Split Problem was introduced by [2] as a challenge to LP-based branch-and-bound approaches. An optimization version of the problem exists but it was originally introduced as a satisfaction problem. An instance consists of $m$ 0-1 equality knapsack constraints on the same $10(m-1)$ variables. Even small instances ($4 \leq m \leq 6$) are surprisingly hard to solve by standard means. We used the generator from [7], whose resulting instances have the same characteristics as those used in [6] and [1]. Table 2 reports the performance of three search heuristics with two levels of consistency on ten instances with $m = 4$. Note that heuristics based on domain size do not apply here.

| consistency | heuristic | backtracks | | | time (sec.) | | |
|---|---|---|---|---|---|---|---|
| | | mean | min | max | mean | min | max |
| domain | LexicoRandom | 245234.2 | 22770 | 689261 | 180.7 | 17.9 | 443.9 |
| | LexicoMaxSD | 93212.6 | 4634 | 248324 | 159.8 | 8.2 | 433.6 |
| | MaxSD | 59870.6 | 7015 | 175596 | 257.3 | 34.5 | 630.4 |
| bounds | LexicoRandom | 5848346.0 | 20671 | 19403712 | 121.3 | 0.4 | 396.9 |
| | LexicoMaxSD | 2463102.0 | 91989 | 8647160 | 116.8 | 4.4 | 400.1 |
| | MaxSD | 12543500.0 | 2701376 | 22010947 | 957.6 | 209.7 | 1664.7 |

**Table 2.** Results of a few search heuristics on ten 4-30 Market Split instances generated from [7].

*With domain consistency.* We observe that the `MaxSD` heuristic, based on the solution density of variable-value pairs, achieves about a four-fold reduction in the average number of backtracks compared to a random heuristic (variable selection is lexicographic but the coefficients of the constraints were randomly generated), but at the expense of slightly higher runtimes. The number of backtracks of `LexicoRandom` is consistent with what was reported in [6] with the same domain consistency algorithm for knapsack constraints. The `LexicoMaxSD` heuristic uses solution density information only to guide value selection. Since fewer solution densities are examined (the choice of variable is fixed), the search heuristic will be faster but probably less accurate as well. Despite a noticeable increase in the average number of backtracks with respect to `MaxSD`, the average runtime improves enough to beat `LexicoRandom`. The random restart strategy was tried in combination with the heuristics but it significantly deteriorated their performance: several instances could not be solved within the one-hour time limit.

*With bounds consistency.* As expected with this weaker level of consistency, the number of backtracks is significantly larger than with domain consistency but most runtimes are reduced as well. The `LexicoMaxSD` heuristic is the fastest overall. `MaxSD` does not perform well here.

### 4.2 Multidimensional Knapsack Problem

This set corresponds to the six `mknap` instances used in [5]. The `mknap1` set from the OR-Library, which are optimization problems, are transformed into satisfaction problems by fixing the objective function to its optimal value, thereby introducing a 0-1 equality knapsack constraint. The other constraints are upper bounded knapsack constraints on the same variables. The instances are of increasing size.

Tables 3 and 4 report the performance of search heuristics on the six instances. Among the heuristics tested, only `LexicoMaxSD` with bounds consistency is able to solve the last instance within an hour. With domain consistency, we note a correlation between increased use of exact solution densities and decreased

| consistency | heuristic | instance (#vars;#constraints) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 6;11 | 15;11 | 20;11 | 28;11 | 39;6 | 50;6 |
| domain | LexicoRandom | 0 | 10 | 157 | 3119 | 85275 | – |
| | LexicoMaxSD | 0 | 2 | 56 | 448 | 39305 | – |
| | MaxSD | 0 | 2 | 40 | 18 | 1438 | – |
| bounds | LexicoRandom | 1 | 66 | 729 | 22052 | 176615 | – |
| | LexicoMaxSD | 0 | 35 | 376 | 16937 | 98993 | 21532762 |
| | MaxSD | 0 | 0 | 3676 | 260952 | – | – |

**Table 3.** Number of backtracks of a few search heuristics on six multidimensional knapsack instances. Instances are labeled by their size: "number of variables; number of constraints". A cutoff time of one hour was used.

| consistency | heuristic | instance (#vars;#constraints) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 6;11 | 15;11 | 20;11 | 28;11 | 39;6 | 50;6 |
| domain | LexicoRandom | 0.0 | 0.2 | 2.2 | 79.8 | 912.7 | – |
| | LexicoMaxSD | 0.0 | 0.2 | 1.9 | 22.8 | 795.0 | – |
| | MaxSD | 0.0 | 0.2 | 1.4 | 2.1 | 57.3 | – |
| bounds | LexicoRandom | 0.0 | 0.0 | 0.0 | 0.9 | 5.9 | – |
| | LexicoMaxSD | 0.0 | 0.0 | 0.1 | 3.1 | 13.4 | 3047.7 |
| | MaxSD | 0.0 | 0.0 | 0.9 | 72.5 | – | – |

**Table 4.** Runtime in seconds of a few search heuristics on six multidimensional knapsack instances. Instances are labeled by their size: "number of variables; number of constraints". A cutoff time of one hour was used.

backtracks and runtimes. Note however that these results are not statistically very significant because there are only three instances of reasonable size – the instances were used because they previously appeared in [5]. It is difficult to compare our results to those because in that paper only bounds consistency was enforced on knapsack constraints. With that same level of consistency, our heuristics do not perform as well.

### 4.3 Magic Square Completion Problem

This very old puzzle is built on a square $n \times n$ grid and asks that we place the first $n^2$ integers in the grid so that each row, column and main diagonal sums up to the same value. A partially filled Magic Square Problem asks for a solution, if one exists. It can be made harder to solve than the traditional version starting from a blank grid. This time we have two types of constraints, $2n+2$ integer knapsack constraints on $n$ variables and one alldifferent constraint on $n^2$ variables. Note that the knapsack constraints have unit coefficients. Each variable ranges over $n^2$ values.

We first generated twenty $9 \times 9$ instances with about 10% of the squares already filled in. Table 5 reports our results. Our heuristics exploiting domain

| consistency | heuristic | backtracks | | | time (sec.) | | | # unsolved |
|---|---|---|---|---|---|---|---|---|
| | | mean | min | max | mean | min | max | instances |
| domain | DomRandom | 9362.4 | 3 | 354346 | 27.1 | 7.2 | 661.3 | 1.8 |
| | DomMaxSD | 694.4 | 2 | 3642 | 23.5 | 20.6 | 28.6 | – |
| | MaxSD | 3095.1 | 2 | 50750 | 27.3 | 21.4 | 36.4 | – |
| bounds | DomRandom | 133931.6 | 43 | 3864972 | 4.4 | 0.0 | 123.1 | 0.4 |
| | DomMaxSD | 266358.0 | 34 | 3159163 | 12.7 | 0.1 | 145.0 | 1.0 |
| | DomMaxSD+ | 1258030.0 | 281 | 11816592 | 94.8 | 0.3 | 838.6 | – |
| | MaxSD | 200574.0 | 34 | 3148707 | 27.0 | 0.1 | 421.2 | 1.0 |
| | MaxSD+ | 161512.0 | 280 | 757915 | 47.8 | 0.2 | 224.3 | 3.0 |

**Table 5.** Results of a few search heuristics on twenty $9 \times 9$ Magic Square completion instances with about 90% holes. A cutoff time of one hour was used.

| consistency | heuristic | backtracks | | | time (sec.) | | | # unsolved |
|---|---|---|---|---|---|---|---|---|
| | | mean | min | max | mean | min | max | instances |
| domain | DomRandom | 15679.0 | 18 | 341084 | 72.8 | 2.2 | 2020.0 | – |
| | DomMaxSD | 2442.9 | 6 | 16014 | 14.6 | 2.3 | 96.6 | – |
| | MaxSD | 3102.6 | 7 | 17969 | 19.6 | 2.6 | 137.4 | – |
| bounds | DomRandom | 856060.1 | 183 | 17754444 | 36.3 | 0.0 | 744.9 | 0.2 |
| | DomMaxSD | 1503090.0 | 1232 | 26527968 | 88.7 | 0.1 | 1565.0 | – |
| | DomMaxSD+ | 236459.0 | 1023 | 1257436 | 20.1 | 0.1 | 111.1 | – |
| | MaxSD | 284006.0 | 1098 | 2741124 | 62.3 | 0.2 | 597.1 | 1.0 |
| | MaxSD+ | 200094.0 | 647 | 1219949 | 84.9 | 0.3 | 402.5 | 1.0 |

**Table 6.** Results of a few search heuristics on twenty $9 \times 9$ Magic Square completion instances with about 50% holes. A cutoff time of one hour was used.

consistency perform well but with a noticeable computational fixed cost probably due to the size of the underlying graph, which impacts the solution density computation. Despite the fact that these instances seem to have many solutions (or maybe because of it) — a `DomRandom` heuristic with bounds consistency solves almost every instance, often very fast — our heuristics based on discrete uniform distributions perform worse than `DomRandom`.

We then generated twenty $9 \times 9$ instances with about half of the squares already filled in. Table 6 reports our results. With domain consistency, all three heuristics solve every instance but our heuristics perform better both in terms of number of backtracks and runtime. Our heuristics with bounds consistency require about two orders of magnitude more backtracks but runtimes that are less than an order of magnitude longer.

### 4.4 Cost-Constrained Rostering Problem

This set was constructed for this paper and is inspired from a rostering context. Here a 25-day schedule is planned for four employees, who each day either work

a two-, three-, five-hour shift, or not at all. Every day, exactly one of each type of shift must be covered. There is an hourly cost for making someone work, which varies both across employees and days. For each employee, the total cost must lie within a certain range. Finally, some employees are unavailable for certain shifts on certain days.

An employee-centered model for this problem has 100 variables in all (one per employee and per day), each with domain $\{0, 2, 3, 5\}$. There are 25 alldifferent constraints on four variables each (one for each day). There are four knapsack constraints on 25 variables each (one for each employee): the integer coefficients corresponding to the hourly costs are drawn uniformly at random from $[0, 9]$. Ten unavailabilities exclude some values from the domains. We consider two variants.

**Upper-bounded costs** In this variant, the total cost for each employee is bounded above by an integer drawn uniformly at random from $[240, 260]$. This translates into upper bounded integer knapsack constraints.

| consistency | heuristic | backtracks | | | time (sec.) | | | # unsolved |
|---|---|---|---|---|---|---|---|---|
| | | mean | min | max | mean | min | max | instances |
| domain | DomRandom | 152607.6 | 0 | 2653226 | 197.3 | 0.3 | 3280.5 | 3.0 |
| | DomMaxSD | 0 | 0 | 0 | 0.6 | 0.5 | 0.7 | – |
| | MaxSD | 0 | 0 | 0 | 0.9 | 0.8 | 1.0 | – |
| bounds | DomRandom | 644552.3 | 0 | 10692243 | 50.8 | 0.0 | 862.4 | 2.0 |
| | DomMaxSD | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 | – |
| | DomMaxSD+ | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 | – |
| | MaxSD | 0 | 0 | 0 | 0.0 | 0.0 | 0.0 | – |
| | MaxSD+ | 0 | 0 | 0 | 0.1 | 0.1 | 0.1 | – |

**Table 7.** Results of a few search heuristics on ten rostering instances with upper bounded costs. A cutoff time of one hour was used.

Table 7 reports the performance of search heuristics on ten feasible instances. These instances are very easy for our heuristics based on solution densities, either exact of approximate: every instance is backtrack-free. The `DomRandom` heuristics leave a few instances unsolved. Refining variable selection by considering domain size over dynamic degree did not help `DomRandom` but adding a random restarts strategy made it possible to solve every instance, with an average runtime under a second in the case of `DomRandom` with bounds consistency.

**Exact costs** In this variant, the total cost for each employee must equal an integer drawn uniformly at random from $[220, 240]$. This translates into equality integer knapsack constraints. Table 8 reports the performance of the search heuristics on ten feasible instances.

| consistency | heuristic | backtracks | | | time (sec.) | | | # unsolved |
|---|---|---|---|---|---|---|---|---|
| | | mean | min | max | mean | min | max | instances |
| domain | DomRandom | – | – | – | – | – | – | 5.8 |
| | DomMaxSD | 7.1 | 0 | 49 | 0.3 | 0.3 | 0.3 | 1.0 |
| | MaxSD | 5.2 | 0 | 40 | 0.4 | 0.4 | 0.5 | – |
| bounds | DomRandom | – | – | – | – | – | – | 5.3 |
| | DomMaxSD | 232.0 | 0 | 1976 | 0.0 | 0.0 | 0.0 | 1.0 |
| | DomMaxSD+ | 29.0 | 0 | 161 | 0.0 | 0.0 | 0.0 | 1.0 |
| | MaxSD | 508.1 | 0 | 4930 | 0.1 | 0.0 | 0.5 | – |
| | MaxSD+ | 89.0 | 2 | 560 | 0.1 | 0.1 | 0.2 | – |

**Table 8.** Results of a few search heuristics on ten rostering instances with exact costs. A cutoff time of one hour was used.

Here heuristic behavior is similar for domain and bounds consistency. Even with smallest-domain variable selection and domain (or bounds) consistency enforced on every constraint of the problem, the behavior of the `DomRandom` heuristic is very erratic, failing to solve more than half of the instances on average and exhibiting backtrack numbers ranging from zero to almost three million. As before, dynamic degree did not help. The addition of a random restarts strategy on top of `DomRandom` helps to solve a few more instances but the overall behavior remains the same. In contrast, our heuristics using solution densities for value selection perform extremely well and show more robustness: low average and maximum number of backtracks. Note however that one instance out of the ten could not be solved by the `DomMaxSD` heuristic given one hour of computing time — using solution densities for variable selection as well appears to be more robust for this problem. A more extensive experiment with 100 similar instances still gave very few backtracks for the `MaxSD` heuristic. For the "bounds consistency" heuristics, there is a noticeable decrease of the number of backtracks when exact means and variances are computed.

## 5 Discussion

We showed how to evaluate the solution density of a set of variables subject to a knapsack constraint. The first method based on domain consistency computes the exact solution density. The second method approximates variable domains with intervals as it is done with bounds consistency. The experiments generally showed a significant advantage of search heuristics based on such information both in the number of backtracks and the computation time. The fact that the increased accuracy of the solution density information is almost always accompanied by a decreased number of backtracks indicates that this is relevant heuristic information.

However the experimental results so far do not clearly indicate which of the two algorithms should be used or even when one dominates the other. It is also unclear yet whether computing the exact mean and variance of a discrete domain

generally helps. We plan to clarify those points in a further investigation. For the moment, we at least measured the relative error made by Algorithm 1 when computing solution densities for the Magic Square Completion Problem. On the instances with 90% holes, we observed a 5% error with exact mean and variance and a 9% error with approximated mean and variance. On the instances with 50% holes, the error was 30% in the first case and 35% in the other.

We haven't attempted here any aggregation of the solution density information from different constraints beyond simply taking the maximum. A true assessment of the potential of such an approach to heuristic search must consider more ways to aggregate. For example, taking the average solution density of a variable-value pair over the constraints in whose scope it is and choosing the pair maximizing that average was tried on the multidimensional knapsack instances and this outperformed the approach of [5].

Note that both solution density algorithms proposed can be easily adapted to lift the restriction of non-negative coefficients and domain values, at the expense of a larger graph in the case of the first algorithm. This means that the scope of this work can be broadened to general linear constraints.

## Acknowledgments

We wish to thank the anonymous referees for their excellent comments and suggestions. Financial support for this research was provided by the Natural Sciences and Engineering Research Council of Canada.

## References

1. K. Aardal, C. A. J. Hurkens, and A. K. Lenstra. Solving a System of Linear Diophantine Equations with Lower and Upper Bounds on the Variables. *Math. Op. Res.*, 25:427–442, 2000.
2. G. Cornuéjols and M. Dawande. A Class of Hard Small 0-1 Programs. *INFORMS J. Computing*, 11:205–210, 1999.
3. G. Pesant. A Regular Language Membership Constraint for Finite Sequences of Variables. In *Proc. CP'04*, pages 482–495. Springer-Verlag LNCS 3258, 2004.
4. G. Pesant. Counting Solutions of CSPs: A Structural Approach. In *Proc. IJCAI'05*, pages 260–265. Professional Book Center, 2005.
5. P. Refalo. Impact-Based Search Strategies for Constraint Programming. In *Proc. CP'04*, pages 557–571. Springer-Verlag LNCS 3258, 2004.
6. M.A. Trick. A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints. *Annals of Operations Research*, 118:73–84, 2003.
7. A. Wassermann. The Feasibility Version of the Market Split Problem. http://did.mat.uni-bayreuth.de/~alfred/marketsplit.html, last consulted on November 10, 2007.
8. A. Zanarini and G. Pesant. Solution Counting Algorithms for Constraint-Centered Search Heuristics. In *Proc. CP'07*, pages 743–757. Springer-Verlag LNCS 4741, 2007.