



Management Science

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

Entropy-Based Optimization of Nonlinear Separable Discrete Decision Models

Yuji Nakagawa, Ross J. W. James, César Rego, Chanaka Edirisinghe

To cite this article:

Yuji Nakagawa, Ross J. W. James, César Rego, Chanaka Edirisinghe (2014) Entropy-Based Optimization of Nonlinear Separable Discrete Decision Models. *Management Science* 60(3):695-707. <http://dx.doi.org/10.1287/mnsc.2013.1772>

Full terms and conditions of use: <http://pubsonline.informs.org/page/terms-and-conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2014, INFORMS

Please scroll down for article—it is on subsequent pages



INFORMS is the largest professional society in the world for professionals in the fields of operations research, management science, and analytics.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

Entropy-Based Optimization of Nonlinear Separable Discrete Decision Models

Yuji Nakagawa

Faculty of Informatics, Kansai University, Ryozenjicho, Takatsuki-shi 569, Japan,
nakagawa@kansai-u.ac.jp

Ross J. W. James

Department of Management, University of Canterbury, Christchurch 8140, New Zealand,
ross.james@canterbury.ac.nz

César Rego

School of Business Administration, University of Mississippi, University, Mississippi 38677,
crego@bus.olemiss.edu

Chanaka Edirisinghe

College of Business Administration, University of Tennessee, Knoxville, Tennessee 37996,
chanaka@utk.edu

This paper develops a new way to help solve difficult linear and nonlinear discrete-optimization decision models more efficiently by introducing a problem-difficulty metric that uses the concept of entropy from information theory. Our entropy metric is employed to devise rules for problem partitioning within an implicit enumeration method, where branching is accomplished based on the subproblem complexity. The only requirement for applying our metric is the availability of (upper) bounds on branching subproblems, which are often computed within most implicit enumeration methods such as branch-and-bound (or cutting-plane-based) methods. Focusing on problems with a relatively small number of constraints, but with a large number of variables, we develop a hybrid partitioning and enumeration solution scheme by combining the entropic approach with the recently developed improved surrogate constraint (ISC) method to produce the new method we call ISCENT. Our computational results indicate that ISCENT can be an order of magnitude more efficient than commercial solvers, such as CPLEX, for convex instances with no more than eight constraints. Furthermore, for nonconvex instances, ISCENT is shown to be significantly more efficient than other standard global solvers.

Keywords: multidimensional nonlinear knapsack; separable discrete optimization; combinatorial optimization; surrogate constraints; problem difficulty estimation; entropy

History: Received April 24, 2011; accepted February 23, 2013, by Dimitris Bertsimas, optimization. Published online in *Articles in Advance* November 18, 2013.

1. Introduction

We propose a new solution concept that may be embedded within an implicit enumeration solution scheme for discrete-optimization problems, both linear and nonlinear, wherein the information theoretic concept of entropy is used to estimate the complexity of solving a potential subproblem, given an available scheme for computing an upper bound. This entropy metric is a proxy for the time required to solve the subproblem, and this information is then used to both partition the problem and determine how the subproblem is to be solved. Information theory has previously been used to establish the complexity of a sorting algorithm (Takaoka 1998); however, to the best of our knowledge, this is the first attempt to use information theory for estimating the difficulty of a

complex combinatorial optimization problem within an optimization framework.

Our proposed approach is general in that the entropy-based partitioning scheme can be used in solution algorithms that use bounds as the primary fathoming criteria, such as the case in branch-and-bound methods, because the theory utilizes upper-bound information. To demonstrate the potential of our approach as a foundation for developing improved computational algorithms, we focus on the class of discrete mathematical optimization models referred to as multidimensional knapsack problems, which are widely known to be of practical importance in several areas of managerial decision making, e.g., capacity planning in computer networks (Gerla and Kleinrock 1977), capital budgeting (Mathur et al. 1983), and production-distribution system design (Elhedhli and Goffin 2005), to name

a few. The essential restriction in this class, compared with general discrete-optimization models, is that the knapsack problem has an objective and constraints that are separable (but possibly nonlinear) in the variables.

More particularly, our focus is on the general multidimensional nonlinear knapsack (MNK) problem, which has broader applications than the multidimensional knapsack problem but is considerably more difficult to solve. Within this domain, we devote attention to both convex and nonconvex MNK instances with a relatively small number of constraints, problems for which the improved surrogate constraint (ISC) method has proven efficient (see Nakagawa 2003). We have coupled our entropic measure approach with the ISC to produce a solution scheme we call the improved surrogate constraint method with entropy (ISCENT). We show that for MNKs with a large number of variables and with up to eight constraints our ISCENT procedure improves solution time by an order of magnitude relative to a solution using standard commercial software such as CPLEX (in the linear and quadratic case) and relative to global solvers such as Bonmin (in the convex case) under the guarantee of exact solutions.

1.1. Additional Relevant Background

It may be noted that separable MNK models can be reformulated as mixed-integer linear programs (MILPs), but with a considerable increase in problem size. However, solving such MILPs using a branch-and-cut approach can become notoriously slow because of difficulties stemming from limitations in memory and processing power (Ralphs 2006). In contrast, our approach works directly on MNKs with convex or nonconvex objectives, thus avoiding size increases due to problem transformations.

It must also be noted that the proposed entropy concept can potentially be embedded within branch-and-cut solution schemes, such as those implemented under commercial software for solving problems with a large number of constraints, for further enhancement of computational efficiency. However, such a computational exercise is outside the scope of the present paper.

The ISC method incorporated within our ISCENT procedure is a branch-and-bound technique for the solution of nonlinear separable discrete-optimization problems and designed to find an exact optimal solution of the problem. Although ISC has been shown to be efficient for problems with a large number of variables, its efficiency deteriorates as the number of constraints in the problem grows. Intriguingly, some large-sized problems are more easily solved by ISC than some small-sized problems, indicating that the problem size, measured in terms of the number of

variables or constraints, is not the sole determinant of problem difficulty. This observation has motivated our development of an entropic measure of problem difficulty, which we then use to partition the problem into easier subproblems. The entropy model proposed in this paper is a modified and improved version of the early approach in Nakagawa (2004), wherein an entropy measure is developed only for linear 0–1 multidimensional knapsack problems. In contrast, our extended entropy model can handle general integer problems with nonlinear (separable) functions. Moreover, we use our proposed entropy measure to perform variable aggregation (as well as to partition the problem) to develop an efficient branch and bound solution scheme. We show through computational experiment that our entropy metric is strongly correlated with the expected CPU time required to solve a subproblem, thus confirming its effectiveness as a measure of problem difficulty.

2. Nonlinear Separable Discrete-Optimization Model

Given n projects and m resources, each project i has $k_i + 1$ levels (items). If project $i \in \{1, 2, \dots, n\}$ is adopted at level $x_i \in \{0, 1, \dots, k_i\}$, then $g_{ji}(x_i)$ is the amount of resource j consumed and $f_i(x_i)$ is the return, where the total resource availability is b_j . The MNK problem can then be stated as

$$\begin{aligned}
 \text{P:} \quad & \max f(\mathbf{x}) = \sum_{i=1}^n f_i(x_i) \\
 & \text{s.t. } g_j(\mathbf{x}) = \sum_{i=1}^n g_{ji}(x_i) \leq b_j \quad (j \in M), \\
 & \quad x_i \in K_i \quad (i \in N),
 \end{aligned}$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is the vector of decision variables, $M = \{1, 2, \dots, m\}$ is a set of constraint indices, $N = \{1, 2, \dots, n\}$ is the set of decision-variable indices, and $K_i = \{0, 1, 2, \dots, k_i\}$ identifies the (allowable) item set for each variable x_i . Without loss of generality, we assume that

$$\begin{aligned}
 \text{(A1)} \quad & f_i(x_i) \geq 0 \quad \text{for } x_i \in K_i, i \in N, \\
 & g_{ji}(x_i) \geq 0 \quad \text{for } x_i \in K_i, i \in N, j \in M.
 \end{aligned}$$

Note that any separable discrete-optimization problem can be transformed into problem P. When $m = 1$, P is simply called a nonlinear knapsack problem. Furthermore, problem P is a 0–1 (linear) knapsack problem when $k_i = 1$ for $i \in N$, provided $f_i(x_i)$ and $g_{ji}(x_i)$ are monotone nondecreasing functions. In the sequel, we focus on problem P in full generality under (A1), but without any monotonicity assumptions.

Downloaded from informs.org by [132.181.203.188] on 16 November 2014, at 15:50 . For personal use only, all rights reserved.

2.1. Historical Perspective on Solution Methods for the Nonlinear Knapsack Problem

The nonlinear knapsack problem and its multidimensional extension have received considerable attention in the literature; see Cooper (1981) for a survey of solution algorithms for the pure integer version of problem P. Marsten and Morin (1978) combine dynamic programming and branch-and-bound techniques to produce a hybrid algorithm for problems with multiple constraints. The multiple-choice knapsack problem (Nauss 1978) is a linearization of the single-constraint nonlinear knapsack problem. Sinha and Zoltners (1979), Armstrong et al. (1983), and Dyer et al. (1984) present algorithms for solving the multiple-choice knapsack problem. Ibaraki and Katoh (1988) consider the resource allocation problem (RAP), which is the minimization of a nonlinear function with a single constraint and bounded integer variables. The separable version of the RAP is a special case of the nonlinear knapsack problem. Bretthauer and Shetty (1995) develop a branch-and-bound algorithm to solve separable problems involving nonlinear resource allocation. Hochbaum (1995) discusses the computational complexity of convex quadratic knapsack problems subject to a single linear constraint.

In recent years, much progress has been made in developing exact methods for 0–1 (linear) knapsack problems and in developing near-optimal (heuristic) methods for variants of knapsack problems. Kellerer et al. (2004) summarize the different types of knapsack problems and various solution methodologies, both heuristic and exact, explored over the years. In particular, the multidimensional 0–1 knapsack problem has received greater attention, possibly because of its increased solution complexity under higher correlations among problem data. This research spans several years, as the following citations show: Gavish and Pirkul (1985), Freville and Plateau (1994), Chu and Beasley (1998), Freville (2004), Freville and Hanafi (2005), Vasquez and Vimont (2005), and Puchinger et al. (2010). Martello et al. (1999) present a combination of two new algorithms, which is shown to outperform all previous methods, for solving a single-constraint 0–1 knapsack problem exactly. Bertsimas and Demir (2002) present an approximate dynamic programming approach for the multidimensional knapsack problem. Martello and Toth (2003) present an exact algorithm for 0–1 knapsack problems with two constraints.

One promising line of research, first proposed by Balas and Zemel (1980) for single-constraint knapsack problems, is the concept of a “core” problem defined by a set of variables that determine how difficult the problem is to solve. Once these core variables are identified, then determining their solution was the

key to solving the entire problem. The core concept was recently generalized to the multiconstraint case by Puchinger et al. (2010) and used as a basis for both optimal and heuristic solution techniques. In a similar vein, Vasquez and Vimont (2005) used variable fixing to obtain “good” families of solutions in their heuristic. The core problem and good solution are both defined on a particular subset of the variables. The entropy-based metric developed in this paper similarly is used to reduce the original problem to those variables for which it is hard to decide whether or not they will appear in an optimal solution. However, although the core variables must be determined prior to solving the problem, in our method variables are eliminated successively one at a time in the course of solving the problem.

To generalize the problem from a single-constraint problem to a multidimensional problem, Puchinger et al. (2010) proposed using surrogate multipliers to combine the constraints to calculate the efficiency measures required for determining the core variables. Our entropy-based approach too relies on a similar mechanism where the constraints of the problem are combined into a single surrogate constraint as a first step. Our computational results show that the surrogate-constraint method is good for problems having approximately eight (or fewer) constraints. To solve problems with larger numbers of constraints, other techniques, such as cutting planes, can be introduced. Combining cutting planes and surrogate constraints is addressed, for instance, in Osorio et al. (2002). Several heuristic approaches have also been proposed in the literature for MNK problems; see e.g., Coit and Smith (1996a, b), Ng and Sancho (2001), Hsieh (2002), and Liang and Smith (2004). Because the MNK problem is a general separable discrete-optimization problem, any efficient solution technique for MNK would be invaluable in all other special cases as well.

In the context of global optimization, the MNK problem is a nonsmooth, discontinuous, nonlinear, nonconvex constrained model with integer variables. When the objective function and all the constraints (except for the integer restrictions) are differentiable nonlinear functions of the decision variables, the problem is called an integer nonlinear smooth optimization program. A number of solvers exist for determining global optimal solutions of smooth nonlinear convex or nonconvex problems; see e.g., Bonmin (Bonami and Lee 2006, Bonami et al. 2005), Baron (Tawarmalani and Sahinidis 2004), and the commercial software Frontline Systems Premium Solver Platform (Frontline Systems 2005). When the objective or any of the constraints are nondifferentiable, the resulting problem is a nonsmooth optimization model, which is the most difficult class of optimization problems to solve.

2.2. The Surrogate-Constraint Framework

As discussed earlier, the proposed entropy-based branch-and-bound scheme relies on the Surrogate Constraint (SC) method as a basic building block, a method that was first introduced by Glover (1965) for solving 0–1 integer programs. In its simplest form, when the SC method is applied to multiconstrained optimization problems, a sequence of single-constraint (knapsack) subproblems is solved. These subproblems are created by replacing the original problem constraints with a single surrogate constraint generated by a weighted combination of the original constraints. Optimal weights, called surrogate multipliers, can be calculated for the constraints using the algorithm proposed by Dyer (1980) or the cut-off polyhedron (COP) method proposed by Nakagawa and Miyazaki (1981) and Nakagawa et al. (1984). Also, a specialized procedure for two-constraint problems is presented by Gavish et al. (1991).

Surrogate-constraint methods, which generally yield stronger relaxations than Lagrangian methods, can encounter a duality gap, implying that an optimal solution to a surrogate-constraint relaxation may fail to produce an optimal solution to the original problem. To overcome this difficulty, Nakagawa (2003) proposed an ISC method for the solution of nonlinear separable discrete-optimization problems, which often provides an exact solution to relatively large-sized problems.

By developing the information-theoretic entropy metric to assess the difficulty of a discrete-optimization model, the ISCENT algorithm in this paper makes a substantial improvement on the pure ISC method. The basic premise of the entropy hypothesis, for instance, in the case of binary problems, is that if assuming either 0 or 1 for a given binary variable provides similar characteristics to the remainder of the problem, then there is an equal likelihood of this binary variable taking on either value at an optimal solution. On the other hand, if there is evidence to the contrary that resulting properties from assuming one value is quite different from the second, then one value may be somewhat more likely than the other to be included in an optimal solution. A difficult problem is typically one that includes many variables that have similar properties with respect to the possible values, and hence, it is difficult to ascertain whether one value is more likely to be a member of an optimal solution. The entropy metric in this paper is designed to objectively assign an index value to the latter difficulty.

In contrast, one competing approach for measuring problem complexity is the percent gap closure (PGC) value developed by Karwan et al. (1987). The authors suggest that problems with a large surrogate gap (i.e., a small PGC value) are more difficult to solve, as

demonstrated in certain problems. Indeed, the PGC value is a relatively good measure for instances generated by using uniform random numbers (Nakagawa 2003). However, our computational experience with more difficult (correlated problem) instances suggests that there are many cases where the ISC method yields an efficient solution even though the associated PGC value is small. It will be shown that our entropy-based difficulty index, on the other hand, remains monotonic with respect to solution time, and thus it is more robust when compared with PGC as a measure of problem difficulty.

The notable advantage of using an entropy-based complexity index is that it can be used to determine if a given subproblem (say, in a branch-and-bound tree) is expected to be computationally tedious and, if so, to partition the problem into smaller problems. Determining which variables to use for partitioning is also a part of the proposed entropic-based approach. In the event it is concluded that the subproblem is of “affordable complexity,” then it can be attempted for direct solution using an exact solver, such as the ISC method. Such an approach, therefore, allows efficient use of computer memory, as opposed to using a direct branch-and-bound method with breadth-first search, which generally consumes a significant amount of memory. In particular, when coupled with the ISC method, our entropy-based approach achieves substantial gains in the solution time relative to standard solution techniques, including the standalone ISC method, as demonstrated by our computational experiments.

3. The Entropy Concept

In information theory, entropy is a measure of the uncertainty associated with a random variable taking a finite number of possible values. In this context, entropy usually refers to the expected value of information contained in a message—a concept introduced by Shannon (1948). The entropy of a random variable is defined in terms of its probability distribution and can be shown to be a good measure of randomness or uncertainty. Solving a discrete-optimization problem can be difficult because of many factors inherent to problem size and characteristics of problem data, which affect different solution techniques differently. However, we take an alternative view by hypothesizing that the difficulty of solving a discrete-optimization problem depends on the difficulty of optimally selecting each variable value in the problem. Based on this hypothesis, we develop the entropy metric to estimate the difficulty for each variable.

Consider a (generic) branch and bound method for solving the MNK problem P , in which subproblems

are typically defined via restrictions placed on the decision variables. Given a current problem at a node (say, the root node), the entropy values of the variables are computed (as presented subsequently), and these, in turn, are used to simplify the problem using the following two major steps:

Step 1: Variable-aggregation process. Combine two variables into a single new variable considering all item combinations of two variables (and eliminating dominated combinations), thus reducing the number of variables in the current problem. The entropy values of the variables are used to determine which two variables are to be combined.

Step 2: Partitioning. The total entropy index of the nodal problem is used to determine whether the problem needs to be partitioned further to reduce its complexity before being attempted by a solver.

The above two steps are elaborated in the sections that follow after the entropic view of problem complexity is developed.

3.1. Entropy of Subproblem Complexity

Consider a variable x_i that can take on values $x_i = k$ for $k \in K_i$. Let the probability that $x_i = k$ appears in the optimal solution be denoted by $p_i(k)$, $k \in K_i$, herein referred to as the *item probability*. The higher the item probability, the lower the uncertainty that the associated variable will be in the optimal solution at that item value; thus, the information content is expressed as $\log(1/p_i(k))$. The so-called *Shannon entropy* for variable x_i is then a measure of the expected information content of not knowing the optimal value of variable x_i , and it is given by

$$h_i := h(x_i) = \sum_{k \in K_i} p_i(k) \log_2 \left(\frac{1}{p_i(k)} \right) \\ = - \sum_{k \in K_i} p_i(k) \log_2(p_i(k)).$$

The value of h_i , herein referred to as the *variable entropy* for x_i , is maximized when the probability $p_i(k)$ is near $1/k_i$ and minimized when $p_i(k)$ is near 0 or 1. The value of h_i is indicative of the difficulty of optimizing the current problem on the variable x_i . That is, it is difficult to determine an optimal value of a variable when the variable entropy is high. Conversely, an optimal value of a variable is relatively easily determined when its variable entropy is small. It is our contention that problems consisting of many variables with high entropy are more difficult to solve. By contrast, problems containing a large number of variables with low levels of variable entropy should be easier to solve. Hence, an objective measure of the overall solution difficulty of the current problem is the *joint entropy* of the variable set x_i , $i \in N$, expressed as

$$\hat{H}(x_1, x_2, \dots, x_n) = \sum_{i \in N} h(x_i) - C(x_1, x_2, \dots, x_n),$$

where $C(x_1, x_2, \dots, x_n)$ is the total correlation of the variables x_i (Watanabe 1960), and it is given by

$$C(x_1, x_2, \dots, x_n) \\ = \sum_{a_1 \in K_1} \sum_{a_2 \in K_2} \dots \sum_{a_n \in K_n} p(x_1 = a_1, \dots, x_n = a_n) \\ \cdot \log_2 \frac{p(x_1 = a_1, \dots, x_n = a_n)}{p_1(a_1), \dots, p_n(a_n)}.$$

The value of C is usually difficult to estimate directly because it depends on the interactions among different variables in the problem. Indeed, $C = 0$ should result if all decision variables take on their respective values independently of each other. Our strategy is to upper bound the joint entropy by using the property of nonnegativity of the total correlation; i.e., $C(x_1, x_2, \dots, x_n) \geq 0$. This leads to the following upper estimate on the joint entropy as the model of problem difficulty, herein referred to as the *problem entropy*, and it is denoted by H :

$$H(x_1, x_2, \dots, x_n) := \sum_{i \in N} h(x_i) \geq \hat{H}(x_1, x_2, \dots, x_n).$$

Therefore, the problem entropy $H(x_1, x_2, \dots, x_n)$ is the sum of all variable-entropy values. As shown in our computational experiments, this problem-entropy metric disregarding the correlation term C still proves effective as a measure of problem difficulty. Although the inclusion of the correlation term may induce further improvements, to the best of our knowledge, there is no plausible way to compute C . The reason is that it involves estimating joint probabilities $p(x_1 = a_1, \dots, x_n = a_n)$ of different variables taking on different item values. This is an interesting and open problem that deserves further investigation as methodologies to determine C may prove worthwhile in improving the above problem-entropy metric.

To compute the entropy metric H , developing a theoretical probability model for $p_i(k)$, $k \in K_i$, is an onerous task in itself because it depends on the characteristics of problem and underlying data. Thus, we resort to an empirical estimation approach, which is presented in the subsequent section based on a class of test problems. This procedure is based on a conditioning event that specifies whether $x_i = k$ is fixed or $x_i \neq k$ in the current problem (at a node of a branch-and-bound tree). To this end, the existence of an upper-bound generation process for the current problem is crucial.

Consider an upper bound $v^{\text{UB}}[\bullet]$ on a given problem \bullet ; i.e., $v^{\text{UB}}[\bullet] \geq v^{\text{Opt}}[\bullet]$, where $v^{\text{Opt}}[\bullet]$ denotes the optimal value of problem \bullet . Then, $f^{\text{UB}} \equiv v^{\text{UB}}[P]$ is an upper bound on problem P , and $v^{\text{UB}}[P: x_i = k]$ denotes an upper bound on problem P with the restriction that the variable x_i is fixed at value k . Noting that $v^{\text{Opt}}[P] \geq v^{\text{Opt}}[P: x_i = k]$ holds, we assume

the upper bound to be consistent in that $v^{UB}[P] \geq v^{UB}[P: x_i = k]$ holds as well. Furthermore, suppose f^{LB} is a lower bound on problem P. Define the normalized difference of the upper bounds, denoted δ_{ik} , by

$$\delta_i(k) := \frac{f^{UB} - v^{UB}[P: x_i = k]}{f^{UB} - f^{LB}}, \quad k \in K_i, i \in N.$$

Note that $v^{UB}[P: x_i = k] \geq f^{LB}$ must hold; otherwise, the branch $x_i = k$ should have been fathomed. Thus, it follows that $0 \leq \delta_i(k) \leq 1$. If $x_i = k$ is optimal, denoted by $x_i^* = k$, then $v^{UB}[P: x_i = k]$ is expected to be larger (than f^{LB}), and thus $\delta_i(k)$ would be smaller. Conversely, observing a larger $\delta_i(k)$ is indicative of a relatively smaller chance of $x_i = k$ being optimal in problem P. This δ -metric is utilized as a conditioning argument in the empirical estimation of item probabilities, and hence, the entropy of a variable.

4. Upper Bounds via Surrogate Constraints and Entropy Estimation

Surrogate-constraint relaxation provides an effective approach to compute an upper bound f^{UB} on the optimal value of problem P, which is an essential cornerstone of our method to estimate problem entropy. Let $\mathbf{u} = (u_1, u_2, \dots, u_m)$ be a vector of nonnegative weights associated with constraints in problem P. Upon taking a row aggregation of the constraints, with \mathbf{u} being the aggregating multipliers, the following surrogate problem $P^S(\mathbf{u})$ is obtained as a relaxation of the original problem P:

$$\begin{aligned} P^S(\mathbf{u}): \quad & \max f(\mathbf{x}) \\ & \text{s.t. } \sum_{j \in M} u_j g_j(\mathbf{x}) \leq \sum_{j \in M} u_j b_j \\ & x_i \in K_i \quad (i \in N), \end{aligned}$$

and thus $v^{Opt}[P] \leq v^{Opt}[P^S(\mathbf{u})]$ for any $\mathbf{u} \geq 0$. Note that $v^{Opt}[P^S(\mathbf{u})] = v^{Opt}[P^S(\lambda \mathbf{u})]$ for any scalar $\lambda > 0$, and thus \mathbf{u} may be normalized. Consequently, the best surrogate upper bound is determined by the surrogate dual problem defined by

$$P^{SD}: \quad \min \{v^{Opt}[P^S(\mathbf{u})]: \mathbf{u} \in \mathbf{U}\},$$

where

$$\mathbf{U} = \left\{ \mathbf{u} \in R^m: \sum_{j=1}^m u_j = 1, \mathbf{u} \geq 0 \right\}.$$

Suppose $\mathbf{u}^* = (u_1^*, u_2^*, \dots, u_m^*)$ denotes an optimal (surrogate) multiplier of P^{SD} . Then, the upper bound f^{UB} on P, is determined by $f^{UB} = v^{Opt}[P^S(\mathbf{u}^*)]$. An optimal multiplier \mathbf{u}^* is obtained by solving the surrogate dual P^{SD} using the efficient algorithms proposed by Dyer (1980), Nakagawa and Miyazaki (1981), or Nakagawa et al. (1984). (Details are omitted here, and the reader is referred to the latter references.)

Let an optimal solution of the surrogate problem $P^S(\mathbf{u}^*)$ be denoted by \mathbf{x}^{SD} . If \mathbf{x}^{SD} is feasible in problem P, then $v^{Opt}[P] = f^{UB} = f(\mathbf{x}^{SD})$ follows and the given problem is solved. However, when \mathbf{x}^{SD} is infeasible in problem P, then a surrogate *duality gap* exists; i.e., $v^{Opt}[P] < f^{UB}$.

An improved upper bound, which may be computationally more expensive to obtain, is expected to provide a more accurate estimate of problem difficulty via the entropy so-computed. In our implementation, we shall follow the modular approach developed in Nakagawa and Iwasaki (1999) to improve the bound f^{UB} .

4.1. Empirical Estimation of Item Probability

Estimation of the item probability $p_i(k)$ is crucial for determining the entropy measure of problem difficulty. The δ metric presented earlier using the preceding upper-bound generating process is used for this purpose. Recall that the magnitude of $\delta_i(k)$ is indicative of the likelihood that $x_i = k$ is optimal in the problem. However, to convert $\delta_i(k)$ to the probability of variable $x_i = k$ in the final solution, $(p_i(k))$, we need a (transformation) distribution, $p_i(k) = F(\delta_i(k))$, which will be empirically estimated. To develop this distribution F , our approach is to use a certain test set of problems of the class under discussion in this paper with optimal solutions that are known. Then, the $\delta_i(k)$ values of these problems are grouped into sets of increasingly more “difficult” variables to estimate the transformation distribution.

A representative set of 30 classic multidimensional 0–1 knapsack benchmark problems presented in Chu and Beasley (1998)¹ was analyzed, and problems with five constraints and 250 binary variables, herein referred to as set A1, are used for the estimation purposes. This test set of problems is divided into three groups of 10 problems each, where each group is characterized by a different *constraint-tightness* ratio γ , which refers to how restrictive the constraint is. The values of the right-hand side are generated as of the sum of constraint coefficients c_{ji} in every knapsack; i.e., $b_j = \gamma \sum_{i=1}^n c_{ji}$. As the constraint becomes tighter, i.e., a smaller tightness ratio, an optimal solution is likely to contain fewer positive values. The three groups have constraint tightness $\gamma = 0.25, 0.5$, and 0.75 , and these problems are referred to by the identifiers 00 ~ 09, 10 ~ 19, and 20 ~ 29, respectively, within the three groups. Moreover, all problems are generated using correlated random numbers, yielding instances that are significantly more difficult to solve than problems generated by independent uniform random numbers. Within each problem group,

¹These test instances are currently available at <http://people.brunel.ac.uk/~mastjib/jeb/info.html> (last accessed October 18, 2013).

Table 1 Relationship Between the Error Function ϕ and the Upper-Bound Ratio δ

$\delta_i(k)$	Proportion of variables in the range where the largest upper bound is generated with a nonoptimal item			Approximated error function $\phi_i(k)$
	Problem number (size $5 \times 250 \times 2$)			
	00 ~ 09	10 ~ 19	20 ~ 29	
0.00 ~	0.5469	0.5517	0.4098	0.4756
0.01 ~	0.28	0.2937	0.3085	0.3030
0.10 ~	0.1161	0.1008	0.1221	0.1113
0.20 ~	0.0421	0.0382	0.049	0.0409
0.30 ~	0.0265	0.0167	0.0187	0.0150
0.40 ~	0.0074	0	0	0.0055
0.50 ~	0	0	0	0.0020
0.60 ~	0	0	0	0.0007
0.70 ~	0	0	0	0.0003
0.80 ~	0	0	0	0.0001
0.90 ~ 1.0	0	0	0	0.0000

δ is calculated for $k \in \{0, 1\}$, $i \in N$, and δ is categorized into 10 subintervals of equal length from the interval $[0, 1]$. For each of these categories and problem groups, we define the error function $\phi(\delta)$ as the proportion of problems in which a nonoptimal item is associated with the largest upper bound. These proportions are presented in Table 1, where a total of 2,500 variables are classified from the 10 sample problem instances in each column.

To illustrate, consider the problem group 00 ~ 09 and the category range 0.00 ~ 0.01 for δ . There were a total of 64 variables with δ values that fell into this category. For 35 of these 64 variables, the smaller upper bound was generated when the variable was set to its *known* optimal value; i.e., for these variables, $v^{\text{UB}}[\text{P}: x_i \neq k^*] > v^{\text{UB}}[\text{P}: x_i = k^*]$. For the remaining 29 variables, the known optimal value produced a larger upper bound; i.e., for these variables, $v^{\text{UB}}[\text{P}: x_i = k^*] \geq v^{\text{UB}}[\text{P}: x_i \neq k^*]$, where k^* is the known optimal value of variable x_i . Thus, the error proportion $\phi(\delta)$ is sample estimated using the fraction $35/64 = 0.5469$.

It is evident from Table 1 that $\phi(\delta)$ is decreasing in δ and the decay of $\phi(\delta)$ in δ is more prominent at higher levels of δ . This justifies the use of an *exponential* function to fit the empirical proportions, denoted by ϕ as well. It is determined that the model

$$\phi(\delta) = 0.5 \times e^{-10.017 * \delta}$$

fits the empirical proportions closely, as shown by the last column of Table 1. The proportion $\phi(\delta)$ may be viewed as an error function that decays exponentially as δ increases. Similar exponential decays (in strength) are commonplace in many natural phenomena, e.g., radioactive substances, chemical reactions,

electric charges, and many enzyme-catalyzed reactions, in these cases, with respect to time.

4.2. Item Probability Transformation Function

Whereas the above empirical analysis was focused on binary problems, we extend the error proportion decay function ϕ (which has a range from 0 to 0.5) to nonbinary (general integer) problems as well. We first convert the error proportions to a likelihood function, $\theta(\delta) = \phi(\delta)/(1 - \phi(\delta))$. Note that $\theta(\delta) = 1$ when $\delta = 0$ and $\theta(\delta) \approx 0$ when $\delta = 1$. For a given variable x_i , and an item $k \in K_i$, let the observed value of normalized difference be $\delta_i(k)$. Let us denote the associated error likelihood by $\theta_i(k) \equiv \theta(\delta_i(k))$. Recall that if $x_i^* = k$, then $\delta_i(k)$ is expected to be smaller with a high probability. Therefore, $\theta_i(k)$ may be interpreted as the conditional probability that $\delta = \delta_i(k)$ is observed given that the optimal solution is $x_i^* = k$; i.e., $\theta_i(k) = \text{Pr}(\delta = \delta_i(k) | x_i^* = k)$. Then, using *Bayes' theorem*, the item probability function is obtained as

$$p_i(k) = \text{Pr}(x_i^* = k | \delta = \delta_i(k)) \\ = \frac{\theta_i(k)}{\sum_{s \in K_i} \theta_i(s)}, \quad \forall k \in K_i, i \in N.$$

4.3. Validation of the Entropy Model

We use the ISCENT algorithm, to be developed in the subsequent sections, to provide a motivation for and justification of the validity of the entropy metric. For this, we employ an empirical approach using 30 sample test problems, each having five constraints and 500 variables, referred to as problem set A2 in §6.1. A personal computer (dual-core i5-2520M CPU 2.5 GHz) was used for these experiments.

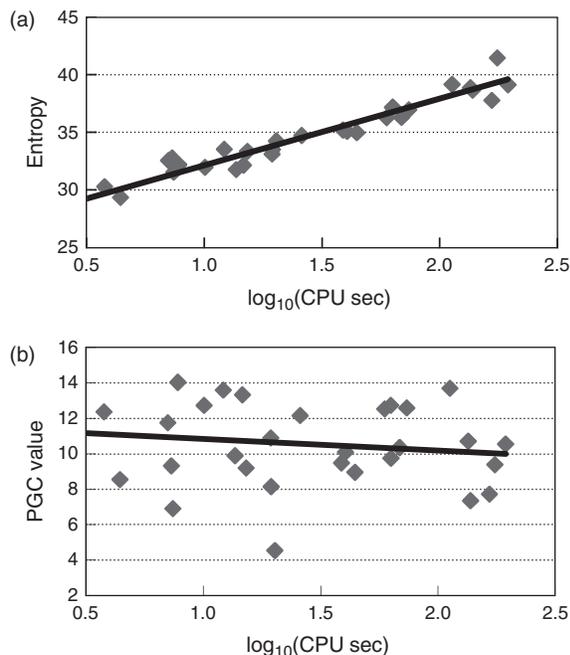
Under the preceding entropy metric of problem difficulty, ISCENT is used to obtain the exact optimum solution of each test problem. The relationship between the entropy H and the (natural) logarithm of CPU time is depicted in Figure 1(a), yielding a correlation coefficient of 0.9663. This indicates a strong association between the entropy metric and the solution time of the problem.

As mentioned previously, an alternative metric of problem difficulty is the PGC proposed by Karwan et al. (1987). Our entropy metric is superior to the PGC metric in the class of problems addressed here. To demonstrate this, we apply the PGC model of problem difficulty as defined by

$$\frac{f^{\text{UB}} - v^{\text{Opt}}[\text{PSD}]}{f^{\text{UB}} - v^{\text{Opt}}[\text{P}]} \times 100,$$

where $v^{\text{Opt}}[\bullet]$ denotes the optimal objective function value of problem \bullet , and PSD is the surrogate dual problem of the original problem P. The upper bound f^{UB} is computed using the same procedure used in the

Figure 1 (a) Solution Time and the Entropy Metric; (b) Solution Time and the PGC Value



entropy metric. The resulting correlation coefficient between the PGC value and the logarithm of the (ISC) CPU time is weak at -0.1525 ; see Figure 1(b). To the best of our knowledge, there are no other metrics (except for the PGC) for estimating problem difficulty for nonlinear separable non-0–1 discrete optimization including the 0–1 knapsack problem.

5. Entropy-Based Improved Surrogate-Constraint Method

The proposed entropy-based method operates on a candidate list, L , of subproblems, initialized with the original problem P . Consider a particular subproblem P^S from the list L , of which the problem entropy is first evaluated. If the problem entropy H falls below a given threshold σ , the problem P^S is judged easy enough to be solved directly, in our case, employing the ISC method. Otherwise, a variable with the maximum variable entropy, h_i , is selected from the variables in problem P^S . The subproblem P^S is then partitioned into several problems by fixing the selected variable x_i at specific item values in the set K_i . The resulting new subproblems are added to the candidate list L . This is referred to as the *problem partitioning* (PP) step within the implicit enumeration scheme.

The above maximum-entropy variable-selection policy is similar in spirit to the score-based branch-selection process implemented in an LP-based branch-and-bound used method in modern MIP solvers (Achterberg et al. 2005). In the latter case, a score based on the change in the objective function of the

LP relaxations of the child subproblems and that of the parent subproblem is used. The variable having the best score is usually selected for the next branch for search. In contrast, ours is based on the entropy metric of problem difficulty.

5.1. Fathoming and Variable Aggregation

The second major step of the implicit enumeration method under the entropic view of problem complexity is the *fathoming and variable-aggregation* (FVA) step in the ISC method. Our approach to FVA is a modification of the modular approach described in Nakagawa and Iwasaki (1999), which is a method for reducing the decision space and number of variables. Our modification is that the variable-entropy measure is utilized as the criterion for selecting which two variables are to be combined (or aggregated) into one. A small variable-entropy value generally signals that the variable’s optimal value can be determined easily. We follow a simple aggregation policy in which the variable having the minimum entropy and the variable having the maximum entropy are combined considering all item combinations of two variables. We refer to this as the *min-max aggregation* policy. After aggregating a sufficient number of variables, *easy* variables disappear from the problem and difficult variables with high entropy remain in the problem. The min-max policy is preferred over a policy of aggregating two variables with maximum entropy because the latter tends to result in more difficult variables that require more memory. Likewise, an argument can be made against aggregating easy variables because this also results in more difficult variables requiring relatively more memory. Although there are other heuristic policies one may consider, for the purposes of this paper, the min-max aggregation policy is implemented. The fathoming part of the FVA step is applied to narrow the decision space (i.e., item space) of the variables of the subproblem P^S . Two fathoming tests are employed:

(i) *Feasibility test*. If the subproblem P^S does not include any feasible solutions, the problem has been fathomed.

(ii) *Bounding test*. If an upper bound of the subproblem P^S is less than the objective function value of the current solution, the problem has been fathomed.

The basic intent in the FVA step is to use fathoming and variable aggregation repeatedly until the problem size is reduced in the decision space and/or in the number of variables to a level that meets a given threshold, or until no further reduction is possible. It is noteworthy that even in the event the PP step is not executed because the problem entropy H falls below the given threshold σ , the FVA step would still be executed based on the variable-entropy values.

Figure 2 Schematic of the ISCENT Method

```

Procedure ISCENT (L, XExact, fExact, σ)
While (L is not empty) do
    PS ← ProblemExtraction (L)
    PS ← FathomingTest (PS)
    δ ← ProblemDifficultyEstimate (PS)
    If (δ < σ)
        x ← ISC (PS)
        If (f(x) > fExact)
            fExact ← f(x);
            xExact ← x;
        Endif
    Else
        i ← FixedVariableSelection (PS)
        Partition PS into several subproblems by setting the
        value of variable xi to each of the allowed values;
        Add the new subproblems obtained to the
        candidate list L;
    Endif
Endwhile
End.
    
```

5.2. The ISCENT Algorithm

The overall solution scheme encompassing entropy-based problem partitioning, fathoming, and variable aggregation, along with ISC-based exact solution of reduced subproblems, is referred to as ISC with ENTropy, or the ISCENT algorithm. The pseudocode for the ISCENT method is presented in Figure 2, using the required component routines that follow.

Definitions of the Required Component Routines:

- i. ProblemExtraction (L)—returns one problem, P^S, out of the problem candidate list L based on a depth-first strategy that preferentially selects the subproblem with the lowest entropy value.
- ii. FathomingTest (P^S)—applies the feasibility and bounding tests on the problem P^S and returns the reduced problem P^S.
- iii. ProblemDifficultyEstimate (P^S)—returns an estimate of the problem difficulty, δ, of the problem P^S, according to the criterion defined by our entropy metric of problem difficulty.
- iv. FixedVariableSelection (P^S)—selects one variable x_i from the subproblem P^S based on the variable with the highest entropy value and returns the variable index, i.
- v. ISC (P^S)—executes the ISC method (Nakagawa 2003) and returns an exact optimal solution, x to the problem P^S.

6. Computational Experience with ISCENT

Our computational tests show that the ISCENT method can solve several important classes of problems more efficiently than commercial software, such as IBM ILOG’s CPLEX V12.2. Although ISCENT can solve all problems to optimality, CPLEX failed to solve certain instances of our test problems because of memory limitations. All computational tests were

carried out on a dual-core i5-2520M CPU 2.5 GHz personal computer with 8 GB of memory (Windows 7, 64-bit OS). ISCENT uses one thread (CPU) and 2 GB of memory, whereas CPLEX uses one thread (CPU) and 8 GB of memory, except for problems of class C1 in which four threads are used. All test problems are characterized with variables having the same number of items (i.e., k_i = κ for all i ∈ N). Therefore, problem sizes are expressed as m × n × κ.

6.1. Test Problem Sets

A test bank of 326 multidimensional (linear and nonlinear) knapsack instances is used for the experiments. Instances are classified into one of three distinct sets (A, B, and C), and within each set, there are several subsets of problems. Set A contains 210 problems consisting of three subsets (A1, A2, and A3) of 0–1 linear knapsack problems from Chu and Beasley (1998). These instances are known to be more difficult to solve than arbitrary instances of similar size. Subsets A2a and A2b have the last one and two constraints removed from set A2, respectively. Subsets A3a and A3b also have the last one and two constraints removed from set A3, respectively.

Problem sets B and C contain non-0–1 (integer-valued) convex and nonconvex instances, respectively. Set B1 contains knapsack problems based on Petersen (1967), in which 11.6% of the coefficients are zero. These instances are generated by using variables restrictions of x_i = 0, 1, ..., or 10 instead of the original restriction x_i = 0 or 1, and the following convex quadratic objective function is used:

$$\min f(\mathbf{x}) = \sum_{i=1}^n c_i (x_i - 10)^2,$$

where c_i are coefficients of objective function of the Petersen (1967) problem. The right-hand sides of the constraints in the instances are set to

$$b_j = \left\lceil \gamma \sum_{i=1}^n a_{ji} k_i / 2 \right\rceil.$$

Then, using γ = 0.60, 0.65, 0.70, ..., 1.55, 20 problem instances are generated. Set B is organized into sets B1 and B2, and set C is organized into sets C1 and C2. Instances in B2 and C1 are generated from Chu and Beasley’s (1998) 0–1 knapsack instances by using variables restrictions of x_i = 0, 1, 2, or 3 instead of the original restriction x_i = 0 or 1. Subset B2 uses the quadratic objective function:

$$\min f(\mathbf{x}) = \sum_{i=1}^n c'_i (x_i - 10)^2,$$

and subset C1 uses the cubic objective function:

$$\max f(\mathbf{x}) = \sum_{i=1}^n c'_i x_i + a'_{1i} x_i^2 + a'_{2i} x_i^3,$$

Table 2 Characteristics of Test Problems

Test set	Number of instances	Size $m \times n \times \kappa$	Type
A1	30	$5 \times 250 \times 2$	Linear
A2	30	$5 \times 500 \times 2$	Linear
A2a	30	$4 \times 500 \times 2$	Linear
A2b	30	$3 \times 500 \times 2$	Linear
A3	30	$10 \times 100 \times 2$	Linear
A3a	30	$9 \times 100 \times 2$	Linear
A3b	30	$8 \times 100 \times 2$	Linear
B1	20	$5 \times 50 \times 11$	Smooth convex quadratic
B2	30	$5 \times 250 \times 4$	Smooth convex quadratic
C1	30	$5 \times 250 \times 4$	Smooth concave cubic
C2	36	$5 \times 250 \times 11$	Nonsmooth nonmonotone

where c'_i , a'_{1i} , and a'_{2i} are the coefficients of the objective function, first constraint, and second constraint of the Chu and Beasley (1998) problem, respectively. Problem sets B and C1 are integer-valued smooth (differentiable) optimization problems. Problem set C2 consists of nonsmooth, nonmonotone, nonlinear knapsack instances. These are generated by amalgamating problem instances with five constraints and 100 variables from Chu and Beasley (1998). Ten problem instances are used to produce one non-monotone MNK problem instance with 11 alternative items for each variable. For example, the coefficients for the items of variable x_i are 0, the i th coefficient from instance 1, i th item instance 2, ..., or i th item of instance 10. This produces three different sets of constraint coefficients. The right-hand side is modified by using 12 constraint tightness ratios 0.1, 0.2, ..., 1.2. The 36 instances generated are derived from correlated random numbers and, therefore, are considerably more difficult than problems generated by independent random numbers. The characteristics of the test bed are summarized in Table 2.

6.2. Computational Analysis

Recall that the entropy threshold (σ) is an algorithmic parameter to determine whether a subproblem should be further partitioned based on problem difficulty. Ideally, the choice of σ should be problem dependent; however, determining this threshold a priori is difficult, and it would require extensive empirical analysis. As such, developing effective procedures for determining the value of σ is left as a topic of future research. For the present computational experiments, a constant threshold is used within each major problem class, where we fixed $\sigma = 35$ for set A and subset C1 and $\sigma = 20$ for set B and subset C2. Our test results based on the chosen values of σ are summarized in this section. (For additional details, see <http://www.res.kutc.kansai-u.ac.jp/~nakagawa/orlib/MS/>, which also contains raw data for all problem sets.)

Problems in set A, despite their relatively small size, are characteristically more difficult to solve than arbitrary instances of similar size because of the inherent correlation among objective and constraint coefficients. For a comparative analysis, we provide results for ISCENT and CPLEX V12.2. Results for subsets A1, A2, and A3 are summarized in Table 3(a). Note that the problem difficulty metric PGC is not used for our analysis because PGC does not work well for our instances (just as in the example illustrated in Figure 1(b)).

Both ISCENT and CPLEX yield the exact optimal solutions for all 90 test instances of subsets A1, A2, and A3. CPLEX required an order of magnitude more computation time than ISCENT in sets A1 and A2. The effectiveness of our method in these subsets is especially noteworthy because these problems with up to five constraints are generally considered to be at the outer limits of computational capabilities of standard SC methods, including ISC (Nakagawa 2003). This underscores the effectiveness of the proposed entropy metric. However, problems in subset A3 have 10 constraints, making them more difficult for SC methods relative to the straightforward use of CPLEX.

To determine the influence of the number of constraints on solution time, we use problem subsets A2a, A2b, A3a, and A3b, derived by removing constraints from sets A2 and A3, as described earlier. As the results in Table 3(b) illustrate, ISCENT appears to outperform CPLEX for problems having eight or fewer constraints.

From these experiments, we conclude that the number of constraints in the problem can have a significant impact on the performance of ISCENT. We also note that instances with more constraints may efficiently be solved if cutting-plane techniques were incorporated into the ISCENT method as they are in CPLEX. Development of methods that combine cutting-plane techniques within ISCENT is a potentially beneficial avenue for future research.

Problem sets B and C are the more difficult instances because they include nonlinearities and nonbinary discrete conditions. As such, we compare ISCENT with major global optimization solvers, such as the global MINLP, LINDOGlobal, Couennet, Bonmin, Baron, and the Interval Global Solver in the Frontline Systems Premium Solver Platform 6.5. Bonmin and CPLEX were found to be much more efficient and effective than the rest; hence, we only report results for Bonmin and CPLEX here, and the results of the remaining solvers are available at <http://www.res.kutc.kansai-u.ac.jp/~nakagawa/orlib/MS/>.

The computational results for sets B and C are summarized in Table 4. Recall that subsets B1 and B2 are

Downloaded from informs.org by [132.181.203.188] on 16 November 2014, at 15:50 . For personal use only, all rights reserved.

Table 3(a) Summary Results for Problem Subsets A1, A2, and A3

Test set	Entropy value (H)			Proportion of problems using PP	CPU time					
					ISCENT			CPLEX		
	Avg.	Min	Max		Avg.	Min	Max	Avg.	Min	Max
A1	26.7	18.5	31.8	0/30	1.9	0.2	8.8	56.7	2.8	261.0
A2	34.6	27.4	41.5	14/30	52.1	2.4	194.5	833.1	58.7	2,414.7
A3	31.4	24.4	36.1	8/30	88.8	3.6	561.0	12.6	1.0	40.9

Table 3(b) Summary Results for Problem Subsets A2a, A2b, A3a, and A3b

Test set	Entropy value (H)			Proportion of problems using PP	CPU time					
					ISCENT			CPLEX		
	Avg.	Min	Max		Avg.	Min	Max	Avg.	Min	Max
A2a	25.2	17.2	31.2	0/30	1.6	0.7	5.2	73.6	4.6	227.3
A2b	17.0	9.1	21.2	0/30	0.5	0.4	0.7	13.9	0.2	58.6
A3a	29.1	21.7	35.2	1/30	22.2	2.1	160.2	10.3	0.7	73.0
A3b	26.4	18.2	32.9	0/30	7.7	1.5	74.3	9.1	0.3	106.2

convex quadratic problems with sizes $5 \times 50 \times 11$ and $5 \times 250 \times 4$, respectively, and B1 problems have 11.6% zero coefficients.

For subset B1, containing only 50 variables, CPLEX is clearly the most efficient solver, requiring less than 2 seconds to solve each of the 20 problems, whereas ISCENT and Bonmin take on average 174.2 and 101.4 seconds, respectively. However, it must be noted that ISCENT's average solution time would have been only 63.9 CPU seconds had it not been for the single-problem instance B1-14, which took 2,269.8 seconds.

The use of cutting planes in CPLEX versus surrogate-constraint relaxations as used in ISCENT, both with the purpose of evaluating the upper (or lower) bound of a subproblem in the branch-and-bound tree, is an essential difference between the two solvers. It is apparent from Table 4 that the successes of the cutting-plane strategies of CPLEX in solving subset B1 do not transfer to subset B2. The CPLEX quadratic solver uses much less memory when using four CPU threads rather than one. The CPLEX quadratic solver does not solve any instance of subset B2 because it

ran out of memory even with 8 GB of RAM. Bonmin cannot find any exact solution within the time limit of one day (86,400 seconds). On the other hand, ISCENT solved all B2 instances exactly in a reasonable amount of CPU time. We contend that the success of ISCENT here is due to its entropy-based problem partitioning and variable aggregation techniques.

As for the nonconvex instances in problem set C, both Bonmin and CPLEX failed to solve these instances to optimality. Subset C1 has cubic concave (differentiable) knapsack instances, which are difficult to solve because of the nonconvexities. Subset C2 contains instances in which both the objective function and the five constraints are nonsmooth and non-monotone. We are unaware of any other solver that can yield exact optimal solutions for problems with nonconvex functions similar to those solved in subset C. All instances in set C are solved to optimality by ISCENT.

The correlation coefficients between the problem difficulty metric (i.e., entropy value) and common logarithm of CPU time for ISCENT and CPLEX for all problem sets are given in Table 5.

Table 4 Summary Results for Problem Sets B and C

Test set	Fraction using PP	CPU time (sec)								
		ISCENT			Bonmin			CPLEX		
		Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
B1	14/20	174.2	0.02	2,269.8	101.4*	15.1*	326.0*	0.6	0.1	1.6
B2	30/30	3,897.9	38.0	17,135.3	>1 day**	>1 day**	>1 day**	>3,258.4***	>2,483.8***	>4,798***
C1	30/30	1,319.5	10.1	4,128.6	—	—	—	—	—	—
C2	34/36	3,276.3	0.1	40,163.1	—	—	—	—	—	—

*Bonmin failed to find an exact optimal solution for three of the 20 instances.

**Bonmin was stopped after one day (86,400 seconds) using one CPU thread and 8 GB of RAM.

***CPLEX caused out-of-memory condition when using four CPU threads and 8 GB of RAM.

Table 5 Correlation Between the Entropy and the Logarithm of CPU Time

Test set	Correlation with entropy	
	ISCENT	CPLEX
A1	0.95	0.91
A2	0.97	0.77
A2a	0.88	0.90
A2b	0.61	0.84
A3	0.94	0.89
A3a	0.89	0.88
A3b	0.82	0.89
B1	0.68	0.40
B2	0.94	—
C1	0.90	—
C2	0.95	—

The CPU time of ISCENT is highly correlated with the entropy metric, except for subset A2b and B1. For ISCENT, the A2b instances have almost the same difficulty, as indicated by the small difference between the minimum and maximum CPU time. For set B1, some of the instances become dramatically easier to solve after partitioning them into smaller problems. For example, instance B1-01 has a large entropy value of 54.5, but this can be solved in 12.3 seconds by ISCENT because constraint coefficients have few digits, 12% zeros, 44% with one digit, 40% with two digits, and 4% with three digits.

The correlation between entropy and CPLEX is relatively good except for the cases where the cutting-plane strategies within CPLEX work well (e.g., subset B1). In passing, we also observe that instances that take more than 1,000 seconds of CPU time have an entropy value of more than 35, but instances with an entropy value of less than 35 take less than 1,000 seconds.

7. Concluding Remarks

It is widely known that problem size is not the only factor that makes an optimization problem difficult to solve. In combinatorial optimization, where the feasible solution space is discrete, it is important to find effective techniques to reduce problem complexity in a manner that makes it possible to combat the effect of the intrinsic combinatorial explosion. Typically, problem size (and especially the number of domain variables) is taken as an indicator of the likely computational complexity (or difficulty) of the problem. In this paper, we propose a new method to estimate problem difficulty based on the concept of variable uncertainty as measured by entropy. The variable uncertainty identifies the degree of difficulty in determining the value a variable should receive in the optimal solution. Problem difficulty is a function of this uncertainty defined over the set of all

problem variables. Analyses carried out on various benchmark problems prove the effectiveness of the proposed technique to estimate problem difficulty, revealing that the underlying entropy metric provides an appropriate foundation for such an estimate.

The success of the proposed method is achieved by embedding the entropy metric within a problem partitioning and variable-aggregation approach that we have joined with the ISC method for solving multi-dimensional nonlinear knapsack problems. More precisely, the gains achieved by our approach are based on three interacting processes: (1) successive partitioning of the problem into smaller problems, induced by the variable of maximum entropy, to mitigate the combinatorial explosion; (2) iterative aggregation of variables based on variable entropy that reduces the problem to a smaller set of “difficult” variables; and (3) employing a specialized procedure (in this case, the ISC method) to solve the problem on a restricted set of variables. Computational testing demonstrates the efficiency of ISCENT for solving MNK problems compared with standard commercial solvers, most notably in the case of nonconvex problems.

There are multiple directions in which our methodology can be extended. First, our entropy metric establishes an upper bound on entropy by ignoring the total correlation term. A nonzero lower bound on the total correlation can improve the entropy metric, thereby resulting in more precise problem-partitioning and variable-aggregation steps and thus a more efficient solution of a given problem. Second, the proposed entropic view of problem difficulty is general enough to embed it within any branch-and-bound algorithm that uses bounds as its primary fathoming test. Development and specialization of the entropy concept within other solver technologies is thus a potential area for future research. Third, because ISCENT’s performance is adversely affected beyond eight or so constraints (because of the embedded surrogate-constraint technique for computing bounds), it is conceivable that more efficient cutting-plane techniques (such as those in CPLEX) might be used to compute more efficient bounds in the presence of a larger number of constraints.

Because computing the entropy metric requires an existing upper-bounding process internal to a solution algorithm, it is a measure of problem difficulty intrinsic to the solution algorithm. For instance, such an entropy metric may be computed for CPLEX, intrinsic to its own bounding computations, for further computational enhancement. In such cases, the key challenge will be to calibrate the entropy model by performing trial runs on a sample set of problem instances to determine the decay constant and, hence, the item probability function of concern. Although this exercise is carried out with relative ease for MNK

problems, it may become more difficult for more general optimization problems.

Acknowledgments

The authors thank the anonymous referees and Fred Glover for their valuable comments and suggestions. Part of this research was financially supported by Kansai University.

References

Achterberg T, Kocha T, Martin A (2005) Branching rules revisited. *Oper. Res. Lett.* 33:42–54.

Armstrong RD, Kung DS, Sinha P, Zoltners AA (1983) A computational study of a multiple-choice knapsack algorithm. *ACM Trans. Math. Software* 9:184–198.

Balas E, Zemel E (1980) An algorithm for large 0–1 knapsack problems. *Oper. Res.* 28:1130–1154.

Bertsimas D, Demir R (2002) An approximate dynamic programming approach to multidimensional knapsack problems. *Management Sci.* 48:550–565.

Bonami P, Lee J (2006) BONMIN users' manual. <https://projects.coin-or.org/Bonmin/>.

Bonami P, Biegler LT, Conn AR, Cornuejols G, Grossmann IE, Laird CD, Lee J et al. (2005) An algorithmic framework for convex mixed integer nonlinear programs. IBM Research Report RC23771, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

Brethauer KM, Shetty B (1995) The nonlinear resource allocation problem. *Oper. Res.* 43:670–683.

Chu PC, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. *J. Heuristics* 4:63–86.

Coit DW, Smith AE (1996a) Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Trans. Reliability* 45:254–260.

Coit DW, Smith AE (1996b) Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS J. Comput.* 8:173–182.

Cooper MW (1981) Survey of methods of pure nonlinear integer programming. *Management Sci.* 27:353–361.

Dyer ME (1980) Calculating surrogate constraints. *Math. Programming* 19:255–278.

Dyer ME, Kayal N, Walker J (1984) A branch and bound algorithm for solving the multiple-choice knapsack problem. *J. Comput. Appl. Math.* 11:231–249.

Elhedhli S, Goffin JL (2005) Efficient production-distribution system design. *Management Sci.* 51:1151–1164.

Freville A (2004) The multidimensional 0–1 knapsack problem: An overview. *Eur. J. Oper. Res.* 155:1–21.

Freville A, Hanafi S (2005) The multidimensional 0–1 knapsack problem: Bounds and computational aspects. *Ann. Oper. Res.* 139:195–227.

Freville A, Plateau G (1994) An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. *Discrete Appl. Math.* 49:189–212.

Frontline Systems (2005) *Premium Solver Platform User Guide* (Frontline Systems, Incline Village, NV). <http://www.solver.com>.

Gavish B, Pirkul H (1985) Efficient algorithms for solving the multi-constraint zero-one knapsack problem to optimality. *Math. Programming* 31:78–105.

Gavish B, Glover F, Pirkul H (1991) Surrogate constraints in integer programming. *J. Inform. Optim. Sci.* 12(2):219–228.

Gerla M, Kleinrock L (1977) On the topological design of distributed computer networks. *IEEE Trans. Comm.* 25:48–60.

Glover F (1965) A multiphase-dual algorithm for the zero-one integer programming problem. *Oper. Res.* 13:879–919.

Hochbaum DS (1995) A nonlinear knapsack problem. *Oper. Res. Lett.* 17:103–110.

Hsieh Y (2002) A linear approximation for redundant reliability problems with multiple component choices. *Comput. Indust. Engrg.* 44:91–103.

Ibaraki T, Katoh N (1988) *Resource Allocation Problems: Algorithmic Approaches* (MIT Press, Cambridge, MA).

Karwan MH, Rardin RL, Sarin S (1987) A new surrogate dual multiplier search procedure. *Naval Res. Log.* 34:431–450.

Kellerer H, Pferschy U, Pisinger D (2004) *Knapsack Problems* (Springer, Berlin).

Liang Y-C, Smith A (2004) An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Trans. Reliability* 53:417–423.

Marsten RE, Morin TL (1978) A hybrid approach to discrete mathematical programming. *Math. Programming* 14:21–40.

Martello S, Toth P (2003) An exact algorithm for the two-constraint 0–1 knapsack problem. *Oper. Res.* 51:826–835.

Martello S, Pisinger D, Toth P (1999) Dynamic programming and strong bounds for the 0–1 knapsack problem. *Management Sci.* 45:414–424.

Mathur K, Salkin HM, Morito S (1983) A branch and search algorithm for a class of nonlinear knapsack problems. *Oper. Res. Lett.* 2:155–160.

Nakagawa Y (2003) An improved surrogate constraints method for separable nonlinear integer programming. *J. Oper. Res. Soc. Japan* 46:145–163.

Nakagawa Y (2004) A difficulty estimation method for multidimensional nonlinear 0–1 knapsack problems using entropy. *IEICE Trans. Fundamentals* J87-A:406–408.

Nakagawa Y, Iwasaki A (1999) Modular approach for solving nonlinear knapsack problems. *IEICE Trans. Fundamentals Electronics* E82-A:1860–1864.

Nakagawa Y, Miyazaki S (1981) Surrogate constraints algorithm for reliability optimization problems with two constraints. *IEEE Trans. Reliability* R-30:175–180.

Nakagawa Y, Hikita M, Kamada H (1984) Surrogate constraints algorithm for reliability optimization problems with multiple constraints. *IEEE Trans. Reliability* R-33:301–305.

Nauss MR (1978) The 0–1 knapsack problem with multiple choice constraints. *Eur. J. Oper. Res.* 2:125–131.

Ng KYK, Sancho NGF (2001) A hybrid dynamic programming depth-first search algorithm with an application to redundancy allocation. *IIE Trans.* 33:1047–1058.

Osorio MA, Glover F, Hammer P (2002) Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions. *Ann. Oper. Res.* 117:71–93.

Petersen CC (1967) Computational experience with variants of the Balas algorithm applied to the selection of R&D projects. *Management Sci.* 13:736–750.

Puchinger J, Raidl G, Pferschy U (2010) The multidimensional knapsack problem: Structure and algorithms. *INFORMS J. Comput.* 22:250–265.

Ralphs TK (2006) Parallel branch and cut. Talbi E-G, ed. *Parallel Combinatorial Optimization* (John Wiley & Sons, Hoboken, NJ), 53–101.

Shannon C (1948) A mathematical theory of communication. *Bell System Tech. J.* 27:379–423.

Sinha P, Zoltners AA (1979) The multiple-choice knapsack problem. *Oper. Res.* 27:503–515.

Takaoka T (1998) A new measure of disorder—Entropy. *Computing Theory '98: Proc. 4th Australasian Theory Symposium (CATS '98)*, (Springer-Verlag, New York), 77–85.

Tawarmalani M, Sahinidis NV (2004) Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Programming, Ser. A* 99:563–591.

Vasquez M, Vimont Y (2005) Improved results on the 01 multidimensional knapsack problem. *Eur. J. Oper. Res.* 165:70–81.

Watanabe S (1960) Information theoretical analysis of multivariate correlation. *IBM J. Res. Development* 4:66–82.