3-1-2002

Empirical Analysis of Various Multi-Dimensional Knapsack Heuristics

Yong Kun Cho

EMPIRICAL ANALYSIS OF VARIOUS MULTI-DIMENSIONAL KNAPSACK HEURISTICS

THESIS

Yong Kun Cho, Captain, Republic of Korea Army

AFIT/GOR/ENS/02-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
8 Mar 2002	Final	Jun 2001 - Mar 2002

Title and Subtitle Empirical Analysis of Various Multi-Dimensional Knapsack Heuristics	**Contract Number**
	Grant Number
	Program Element Number
Author(s) Capt Yong Kun Cho, Repulic of Korea Army	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Bldg 640 WPAFB, OH 45433-7765	**Performing Organization Report Number** AFIT/GOR/ENS/02-04
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)

Distribution/Availability Statement
Approved for public release, distribution unlimited

Supplementary Notes

Abstract
Since the multidimensional knapsack problems are NP-hard problems, the exact solutions of knapsack problems often need excessive computing time and storage space. Thus, heuristic approaches are more practical for multidimensional knapsack problems as problems get large. This thesis presents the results of an empirical study of the performance of heuristic solution procedures based on the coefficients correlation structures and constraint slackness settings. In this thesis, the three representative greedy heuristics, Toyoda, Senju and Toyoda, and Loulou and Michaelides methods, are studied. The purpose of this thesis is to explore which heuristic of the three representative greedy heuristics perform best under certain combination of conditions between constraint slackness and correlation structures. This thesis examines three heuristics over 1120 problems which are all 2KPs with 100 variables created by four constraint slackness settings and 45 feasible correlation structures. Then we analyze why the best heuristic behaves as it does as a function of problem characteristic. The last chapter presents two new heuristics using knowledge gained in the study. When these new heuristics are competitively tested against the three original heuristics, the results show their better performances.

Subject Terms
Multi-Dimensional Knapsack Problem, Heuristic, Computational Testing

Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU
Number of Pages 92	

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U. S. Government.

AFIT/GOR/ENS/02-04

EMPIRICAL ANALYSIS OF VARIOUS MULTI-DIMENSIONAL KNAPSACK HEURISTICS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Yong Kun Cho, BS

Captain, Republic of Korea Army

March 2002

AFIT/GOR/ENS/02-04

EMPIRICAL ANALYSIS OF VARIOUS

MULTI-DIMENSIONAL KNAPSACK HEURISTICS

Yong Kun Cho, B.S.
Captain, Republic of Korea Army

Approved:

Raymond R. Hill, Lt Col, USAF (Advisor)	date
James T. Moore, Ph.D. (Reader)	date

Acknowledgments

First and foremost, I'd like to express my deepest gratitude to my thesis advisor, Lt. Col Raymond R. Hill, for taking the challenge of guiding me through this thesis effort. Without his guidance and endless support, I would not have dreamed of coming to such a state of accomplishment. The experience, insight and knowledge he has shared with me are invaluable and well appreciated. I would also like to thank Dr. James T. Moore for his interest and support.

In addition, I am indebted to the Korean Army for providing me with this wonderful opportunity of Master's degree program that has broadened my knowledge as well as stimulated my understanding of military technology. Hopefully, all this will help me better serve my country in the future.

Most importantly, I'm forever thankful to my wife and son for being there with love, support and understanding. I also feel grateful for my parents and in-law's in Korea for their support and encouragement. Thank you everyone for your patience, endurance, and strong belief in me.

Yong Kun Cho

Table of Contents

Page

Page

List of Figures

Page

List of Tables

Page

Abstract

Since the multidimensional knapsack problems are NP-hard problems, the exact solutions of knapsack problems often need excessive computing time and storage space. Thus, heuristic approaches are more practical for multidimensional knapsack problems as problems get large. This thesis presents the results of an empirical study of the performance of heuristic solution procedures based on the coefficients correlation structures and constraint slackness settings. In this thesis, the three representative greedy heuristics, Toyoda, Senju and Toyoda, and Loulou and Michaelides' methods, are studied. The purpose of this research is to explore which heuristic of the three representative greedy heuristics performs best under certain combinations of conditions between constraint slackness and correlation structures. This thesis examines three heuristics over 1120 problems which are all the two-dimensional knapsack problems (2KPs) with 100 variables created by four constraint slackness settings and 45 feasible correlation structures. Then we analyze why the best heuristic behaves as it does as a function of problem characteristics. Finally we present two new heuristics using knowledge gained in the study. When these new heuristics are competitively tested against the three representative greedy heuristics, the results show the new heuristics perform better.

EMPIRICAL ANALYSIS OF VARIOUS

MULTI-DIMENSIONAL KNAPSACK HEURISTICS

I. Introduction

1.1 Motivation

Obtaining an exact solution to an integer programming problem in real practice is sometimes less practical in comparison to an easily computed method of acquiring near-optimal solutions via heuristics. As problems get large, exact solutions often need excessive computing time and storage space. Many times, these large problems are merely estimates of reality so an optimal solution does not have much meaning. Considering the imprecision of real-world problem data, and that a precise solution in reality may be meaningless, obtaining a near-optimal solution in a reasonable running time may better satisfy a practitioner in the real world.

The knapsack problem has wide application in many areas. Its general form, the multidimensional knapsack problem (MKP), has frequently been used to model various decision-making processes such as resource-allocation, cargo loading, capital budgeting and cutting stock problems. As with other combinatorial problems, computation time increases rapidly with problem size. Many authors have developed heuristic methods to solve MKPs. Greedy algorithms are commonly used and often yield good solutions. These approaches vary in how they treat the problem and select items for inclusion in the knapsack. Little, if anything, has been done to understand how heuristic differences affect performance. This thesis provides empirical analyses of heuristics for

multidimensional knapsack problems and examines how various heuristics function according to particular test problem characteristics.

1.2 Background

1.2.1 The 0 – 1 Knapsack Problem

Suppose there are n projects. The jth project, $j = 1,\ldots, n$, has a cost a_j and a value c_j. A project is either picked or rejected. There is a resource limit of b available for which the projects compete. The problem of choosing a subset of projects to maximize the sum of the values while not exceeding the resource constraint is the 0 – 1 knapsack problem,

$$\max\left\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \le b,\ x \in (0,1)\right\}. \tag{1}$$

This problem is called the knapsack problem because of the analogy to a hiker's problem of deciding what should be put in a knapsack given a weight or volume limitation on how much can be carried. In general, problems of this sort may have m constraints. We then refer to the problem as the multidimensional knapsack problem (MKP) (Nemhauser and Wolsey, 1988). With the MKP, project selection must simultaneously satisfy all m constraints.

1.2.2 The multidimensional knapsack problem (MKP)

The MKP is a 0-1 programming problem of the following form:

Maximize

$$Z = \sum_{j=1}^{n} c_j x_j \tag{2}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, 2, \ldots, m \tag{3}$$

$$x_j = 0 \ \ or \ \ 1 \qquad\qquad j = 1, 2, \ldots, n \tag{4}$$

where $c_j > 0$ and all $a_{ij} \geq 0$. Additionally at least one $a_{ij} > 0$ for each j. A special case of the MKP is the two-dimensional knapsack problem (2KP), where $m = 2$.

1.3 Problem

Many authors have developed approximate heuristic methods for MKPs (Senju and Toyoda, 1968; Toyoda, 1975; Loulou and Michaelides, 1979; Pirkul, 1987). Most competitively test their heuristic against other heuristics or against some common problem set. Few ever focus on why their approach does well on some problems but not so well on other problems. In short, little has been done to determine why a heuristic does well.

1.4 Objective

The objective of this thesis is to understand what makes a "best" heuristic. In other words, why does the best heuristic work well and why do the poorer heuristics not work so well. This is difficult to answer since performance varies by type of problem. We will focus on constraint tightness, correlation between objective function and constraints, and interconstraint correlation. This is done through the generalizing of test problems, coding heuristic methods (Senju and Toyoda, Toyoda, and Loulou and Michaelides method), running all computer algorithms against a common test set, and analyzing the results. The comparison of these three different methods on generalized

test problems is based on a relative error measure. The relative error is defined as $(Z_{OPT} - Z_{heu})/\ Z_{OPT}$ where Z_{heu} is the heuristic objective function value and Z_{OPT} is the optimal, or best known objective function value for the problem. The minimum relative error gives a better solution as that solution is closer to the optimal solution. Raw objective function values are also used when compiling counts of best performing heuristic.

1.5 Overview

Chapter 2 provides relevant background information. It traces an overview of empirical analyses of heuristics and those concepts that apply to the empirical analysis of algorithms. Chapter 3 then describes how to examine the three representative greedy heuristics, how to analyze comparative data and how to compare heuristic performance. Chapter 4 presents the results of an empirical study of the three representative greedy heuristics. Next, Chapter 5 presents two new heuristics and their results. Finally, Chapter 6 summarizes this work's findings, outlines contributions of this work, and identifies areas for further research.

II. Review of Related Literature

2.1 Introduction

The objective of this literature review is to provide an overview of empirical analyses of heuristics and those concepts that apply to the empirical analysis of algorithms. Additionally, heuristics for multi-dimensional knapsack problems are discussed.

2.2 An Empirical Science

Performance of algorithms may be analyzed in two ways. One is to analyze performance analytically relying on the methods of deductive mathematics. The other is to analyze performance empirically using computational experiments. The former, mathematical, methods are further developed than the empirical science. However, the mathematic results do not usually indicate how an algorithm is going to perform on real problems. If we want to know how an algorithm works on typical problems, computational experiments give much more insight into heuristic performance.

Hooker (1994) suggested, "Empirical science involves theory." Through empirical analysis, we can determine how algorithms work and why algorithms perform well or poorly. Thus, we should analyze algorithms to gain insight into theory.

Hooker said, " An empirical science of algorithms would immediately sidestep several of the problems that beset a purely deductive science." Hooker supports this by stating that an empirical science

- does not rely on proving hard worst-case and average-case theorems;
- unlike worst-case analysis, can focus on typical problems;

- unlike average-case analysis, need not restrict itself to a simple and unrealistic distribution of random problems;
- can finesse the issue of characterizing a typical class of problems

(Hooker, 1994).

2.3 Empirical analysis of heuristics

An efficient heuristic provides a correct solution, not necessarily exact or optimal, in a reasonable amount of time and resources. Heuristics are generally efficient in terms of solution quality, computer resource requirements and computer solving time.

Complexity theory classifies problems according to their solution difficulty. Many problems are easy to solve as there exists a provably polynomial time algorithm for the problem where polynomial means algorithm run time is some polynomial function of problem size. Many other problems are difficult to solve as there is no provable polynomial time algorithm for the problems. These are referred to as NP problems.

NP Problems are very difficult to solve and sometimes, impossible to solve in reasonable time. Heuristics, however, offer a viable alternative. With computing power increasing, heuristics and in particular local search, have come into their own in the optimization world. The next section introduces knapsack and greedy MKP heuristics. We leave local search heuristics for the MKP as a future research area.

2.3.1 Knapsack Problem.

The objective of the knapsack problem is to maximize the value of the items selected without exceeding the resource limits.

$$\max\left\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \le b,\ x \in (0,1)\right\} \qquad (1)$$

To solve a knapsack problem using a heuristic algorithm, let x_j be a project of value c_j, and resource cost a_j. The importance of the project can be determined by considering the ratio c_j/a_j. Sequence the projects so the following holds,

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \ldots \geq \frac{c_n}{a_n} \tag{2}$$

and, considering each item in non-increasing value-resource ratio, add as many items as possible, into the knapsack until it is full.

2.3.2 Greedy algorithms for multi-dimensional knapsack problems

The MKP is encountered when one has to decide how to choose projects to satisfy multiple resource constraints. Many effective solution procedures for the MKP have been developed. Each procedure uses some penalty cost (v_j) to quantify the relative worth of the project.

Toyoda (1975) defined an effective gradient as a means to find a good approximate solution to the MKP. Toyoda's effective gradient is a measure of element worth per unit cost in terms of all resources used. Toyoda first normalized each constraint, so all right-hand side (RHS) values were 1. His penalty cost is defined as follows:

$$v_j = \sum_{i=1}^{m} \frac{a_{ij} b_i^0}{\sqrt{\sum_{i=1}^{m} (b_i^0)^2}} \tag{3}$$

where b_i^0 is resource used in constraint i, and $0 \leq b_i^0 \leq 1$ for $i = 1,\ldots, m$ where m is the number of constraints. Toyoda's method starts with an empty knapsack and then adds the element with the highest scoring effective gradient until a knapsack constraint is met.

The value v_j is updated at each iteration. It is called a primal effective gradient method because the solution is always feasible. An overview of this approach is as follows:

Step 1: Start with all items designated as not contained in the knapsacks.

Step 2: Compute the effective gradient for each candidate element not currently in the knapsack and feasible: $G_j = c_j / v_j$.

Step 3: Order the elements in descending order according to their effective gradient measures.

Step 4: Add highest scoring element to knapsacks, retaining problem feasibility.

Senju and Toyoda's (1968) method is very similar to Toyoda's method. However, their approach starts with all items designated as contained in the knapsack. The heuristic then drops projects (x_j) according to ascending order of a dual effective gradient until feasibility is achieved. Their approach is a two-pass algorithm since the authors realized the computed gradients might cause the algorithm to "over shoot" the feasibility target. Thus, once feasible, the Senju – Toyoda method will attempt to restore some items previously dropped. To define their penalty cost (v_j), let a_{ij} be constraint coefficient of jth constraint, Then, $P_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\}$ is the vector of resource costs for element i of n elements. Let $R = \{\sum_{j=1}^{m} a_{ij}, i = 1, \ldots, n\}$ be the vector of total resource consumed in each of m resource constraints, and $L = \{b_1, b_2, \ldots, b_m\}$ be the vector of right- hand side coefficients and finally slack variables: $S = R - L$.

Consequently, a dual effective gradient is: $G_i = c_i\left(\frac{|S|}{P_i \cdot S}\right)$. An overview of this dual approach is as follows:

Step 1: Start with all items designated as contained in the knapsacks.

Step 2: Compute an effective gradient for each element.

Step 3: Order the elements in ascending order according to their effective gradient measures.

Step 4: Drop elements from lowest effective gradient measure until feasibility with respect to all constraints is achieved.

Step 5: Re-consider dropped elements for inclusion if all constraints have resources remaining.

Loulou and Michaelides' approach (1979) expands on Toyoda's (1975) approach. If two candidate projects have an equal v_j, it is more advantageous to select first that project which consumes less resource. They proposed four different ways to select the penalty factor v_j. To calculate the effective gradient, three important concepts should be introduced.

$DA_i + a_{ij}$: total consumption of resource *i* if x_j is added to the current solution where DA_i is an amount of resource consumed so far.

$1 - (DA_i + a_{ij})$: amount of resource *i* remaining if x_j is added to current solution

$\sum_{k \in SC} a_{ik} - a_{ij}$: future potential demand for resource *i* if x_j is added to current solution where *SC* is a set of candidate variables.

The first choice of penalty factor v_j is then:

$$V_j = Max_{i=1,\ldots,m} \{(DA_i + a_{ij})\left(\sum_{k \in SC} a_{ik} - a_{ij}\right) /(1 - DA_i - a_{ij}) \}. \quad (4)$$

The second method chooses penalty factor v_j to decrease the importance of the ratio by taking its square root:

$$V_j = Max_{i=1,\ldots,m} \{(DA_i + a_{ij})\left(\sum_{k \in SC} a_{ik} - a_{ij}\right)^{1/2} /(1 - DA_i - a_{ij})^{1/2}\}. \quad (5)$$

Third and fourth methods are modified from the first and second methods, respectively. These approaches use v_j until Max$_i$ DA_i becomes "close enough" to 1 and, from then on, select projects according to their profits alone. This modification is called a switch. The switch is actuated when some resource becomes so scarce as to suggest that the algorithm is close to terminating.

Pirkul (1987) and Glover(1977) use a multiplier method and surrogate constraints to transform the MKP into a knapsack problem whose solution provides a bound to the original MKP. A surrogate constraint is an inequality implied by the constraints of an integer programming problem (Glover, 1968). A surrogate constraint is formed using a non-negative linear combination of the constraints. The surrogate problem is defined as follows:

$$\begin{aligned} &Z_s(\mu) = \max\ c\,x \\ &St.\ (\mu A)x \le \mu b \\ &x_j \in \{0,1\}\ \ \forall\ j \in N \end{aligned} \tag{6}$$

where μ is positive multiplier vector of size m. The best bound using this scheme is determined by locating a set of multipliers μ* such that

$$Z_s(\mu^*) = \underset{\mu}{Min}\ Z_s(\mu) \tag{7}$$

If μ* is known, then bounds from $Z_s(\mu^*)$ are better than bounds from both a LP-relaxation and a Lagrangian relaxation (Glover, 1968). Notice that problem (6) is simply a 0 – 1 knapsack problem of the form in equation (1).

Pirkul (1987) solves a series of continuous knapsack problems while conducting a single dimensional search process for each μ_i. Alternatively, he could have used the dual

variables associated with each constraint in the linear programming relaxation. For Pirkul's approach, (1) determine a set of surrogate multipliers using dual variables from the relaxation. Then (2) calculate $C_j/(\mu A)_j$ ratios, re-indexing decision variables according to the decreasing order of these ratios. And (3), sequentially fix variables to 1, considered in index order, retaining problem feasibility. Denote this solution X^0 and define $S_1 = \{j \mid x_j = 1\}$ as the set of variables set to 1. Finally, (4) for each $j \in S_1$ set $x_j = 0$ and repeat (3) to define a new feasible solution that temporarily excludes an x_j from consideration.

Osorio *et al*. (2002) improved on Glover's approach by introducing cutting and surrogate analysis. They fix some variables to zero and separate the rest into two groups, those that tend to be zero and those that tend to be one in an optimal solution. Using an initial feasible solution, they generate logic cuts based on analysis before solving the problem using branch-and-bound. The dual surrogate constraint provides a useful relaxation of the constraint set, and can be paired with the objective function. The resulting surrogate has the following property:

$$\sum (\sum_{j \in N} \sum_{i \in M+N} u_i a_{ij}) x_j \leq \sum_{i \in M+N} u_i b_i \tag{8}$$

where u is the surrogate multiplier vector. Since $\sum u_i b_i$ is the continuous LP relaxed solution, this is the current upper bound (UB).

Now, constraint pairing is,

$$\sum_{j \in N} c_j x_j \geq LB \tag{9}$$

$$\sum (\sum_{j \in N} \sum_{i \in M+N} u_i a_{ij}) x_j \leq UB \tag{10}$$

Defining S as a set that contains the indices of x variables whose values in the relaxed LP solution are equal to zero, and making $\sum_{i \in M+N} u_j a_{ij} = s_j$, the resulting combined constraint can be expressed as

$$\sum_{j \in S} (s_j - c_j) x_j \leq UB - LB . \tag{11}$$

Constraint (11) makes the bounds on the components of x stronger. Apparently, the value of $s_j - c_j$ is larger than the value of $UB - LB$, the corresponding x_j must be zero. UB cannot be changed because it is the solution from an LP relaxation of the problem. Thus, if we get a better LB, we can increase the number of integer variables fixed to zero. This decreases the number of variables to be considered in the branch-and-bound algorithm.

2.3.3 Experimental comparison of heuristics for the knapsack problem

General greedy algorithms for MKPs usually give different answers based on the problem characteristics. Zanakis (1977) compared three heuristic methods (Senju-Toyoda (1968), Kochenberger *et al*. (1974), and Hillier (1969)). For the comparison, he created a set of randomly generated 0-1 test problems with nonnegative coefficients and also used benchmark test problems. Zanakis controlled the number of variables (V), (20, 60 ,100, 200, 500 and 1000), the number of constraints(C), (20, 60, 100 and 1000), and the degree of constraint slackness (30%, 50%, and 90%) in the test problems. He measured computer running time, error and relative error on the test problems. Zanakis results suggest all three methods have solution times that increase linearly up to 40-50 variables and 200 constraints but exponentially thereafter, and much faster with respect to the number of variables (V) than constraints (C). Hillier's (1969) algorithm was the most

accurate but much slower than the other two. Kochenberger's *et al*. (1974) heuristic was the fastest of the three with tight constraints and the most accurate with loose constraints. In general the Senju-Toyoda (1968) algorithm was the fastest, but least accurate on small and medium size problem. Therefore, Zanakis suggested selecting the best heuristics based on the problem characteristics. Loulou and Michaelides made a similar suggestion based on their research results (1979).

Fox and Nachtsheim (1990) evaluated six greedy selection rules on zero-one knapsack problems. They used six rules (I through VI) according to how to choose the penalty factor (w_j). The importance of the project (g_j) can be said to be $g_j = c_j / \sum w_i a_{ij}$ where w_j represents a " weight" assigned to constraint i reflecting its "importance" or the relative "scarcity of resource i ". A greedy algorithm selects x_i, and sets it to 1 according to the largest g_j. Rules I, II and III are based on Fox and Scudder (1985). For Rule I, let w_i^p and b_i^p be "the weight assigned to constraint i and the amount of resource i remaining just prior to the pth application", and let $b_{\min}^p = \min_{i \in C} \{b_i^p\}$ where C is the set of constraints. Note that $b_{\min}^p > 0$. Rule I is as follows:

$$w_i^p = \begin{cases} 1, & if \quad b_i^p = b_{\min}^p \\ 0, & Otherwise \end{cases} \tag{12}$$

Rules II and III are modified from Rule I. A constraint tightness consideration is introduced in these rules. Let s_i^p be the tightness of constraint i prior to the pth application. In Rule II, $s_i^p = b_i^p / \sum_{j \in V} a_{ij}$, and in Rule III, $s_i^p = b_i^p - \sum_{j \in V} a_{ij}$ where V is the set of indices of variables that are possible candidates for setting to one.

The weights used are as follows:

$$w_i^p = \begin{cases} 1, & if \quad s_i^p = s_{\min}^p \\ 0, & Otherwise \end{cases} \tag{13}$$

for all $i \in C$, where $s_{\min}^p = \min_{i \in C}\{s_i^p\}$.

Rule IV is known as simple greedy in that all constraints are regarded as equally important and are given an equal penalty factor. Rule V and VI are based on Toyoda (1975). For their empirical analysis, Fox and Nachtsheim varied four parameters on 1440 randomly generated test problems. Their parameters were number of variables, number of constraints, constraint matrix density, and slackness. They measured the average relative efficiencies and the average rank of the objective function among all six rules. They suggested that Rule IV seems to be the best algorithm in terms of relative efficiencies and rank of the objective function. However, in the mixed slackness (tightness value $s_i = 0.3$ with probability 0.5 and $s_i = 0.7$ with probability 0.5) problems, Rule I, from Fox and Scudder (1985) was superior because Rule IV uses the same importance and same weight. Also, they concluded, " The simplest rule is the best, except when the constraints exhibit mixed slackness."

2.3.4 Studies into effects of correlation on solution procedure performance

Some MKPs can be quickly solved even if n is very large, while other problems cannot be easily solved for n equal to a few hundred. One reason may be that the correlation between objective function and each set of constraint coefficients, and the correlation between constraint coefficients effects solution procedure performance. Many authors developed their randomly generated data set to verify their algorithm, but few have actually studied the effects of correlation among the test problem coefficients.

Martello and Toth (1988, 1997) devised three classes of correlation to check computational performance for knapsack problems:

Uncorrelated: w_j uniformly random in [1, a],
p_j uniformly random in [1, a],

Weakly Correlated: w_j uniformly random in [1, a],
p_j uniformly random in [w_j - δ, w_j + δ]

Strongly Correlated: w_j uniformly random in [1, a],
$p_j = w_j + \delta$

where p_j = profit of item j, and w_j = weight of item j given n items, and a and δ are prefixed constants (Martello and Toth, 1997). These induction strategies have been widely used in empirical studies despite the fact that the weakly correlated scheme produced coefficients correlated to a value of 0.98.

Martello and Toth (1988) conducted experiments with their exact algorithm MT2 solving problems with each correlation level with up to more than 100,000 variables. They reported uncorrelated and weakly correlated instances were easily solved. However the strongly correlated instances were very difficult to solve. They could be solved for small number of variables and constraints, using a dynamic programming algorithm that, however, would not work on larger instances due to excessive space and time requirements. In short, the results of Martello and Toth for the zero-one knapsack problem indicate that problems with nearly perfectly positive correlated processing times and weights are significantly harder than the uncorrelated problems.

Hill and Reilly (2000) measured how the coefficient correlation structure affects solution performance using randomly-generated test sets. For test sets, they controlled

these problem generation parameters: correlation measure (Pearson or Spearman), correlation structure, and the constraint slackness:

Let $A^1 \sim U\{1, 2, \ldots, 40\}$ be the random variable representing the values of coefficients in the first constraint, let $A^2 \sim U\{1, 2, \ldots, 15\}$ be the random variable representing the values of the coefficient in the second constraint, and let $C \sim U\{1, 2, \ldots, 100\}$ be the random variable representing the values of the objective function coefficients in 2KP. The three correlation terms in the correlation structure of 2KP are ρ_{CA^1}, ρ_{CA^2}, and $\rho_{A^1A^2}$ with $p = (\rho_{CA^1}, \rho_{CA^2}, \rho_{A^1A^2})$. A " slackness" measure for constant i, S_i, is defined as the ratio of the right-hand side coefficient in constraint i to the sum of the coefficients in that constraint. Two levels of slackness are examined in this study: $S_i = 0.30, 0.70$, $i = 1, 2$. Each of the four possible setting of S_1 and S_2 is referred to as a constraint slackness setting (Hill and Reilly, 2000).

For their study, they used CPLEX and Toyoda (1975) as solution methods. Their goal was to investigate how problem structure effects solution procedure performance by either algorithm (CPLEX) or heuristic (TOYODA). They measured the number of nodes for CPLEX performance and the relative error for the TOYODA performance. Between Pearson and Spearman test problems, Spearman correlation problems were harder. For correlation structure, they found that the difficult problems requiring more nodes have larger differences between ρ_{CA^1}, ρ_{CA^2}, and $\rho_{A^1A^2}$ in CPLEX. The negative values of $\rho_{A^1A^2}$ yields the harder problems for TOYODA. Interestingly, the challenging problems for CPLEX were easy for TOYODA as it found optimal solutions. For constraint slackness, tight constraints provide more challenging problems for both CPLEX and

TOYODA. The interaction between correlation structure and constraint slackness is that tighter constraints and constraint coefficients with a wider range of values produce more difficult problems. However, in CPLEX, positive interconstraint correlation usually yields more simple problems. For TOYODA, tight constraints and the negative value of $\rho_{A^1A^2}$ make problems harder to solve. Their results indicate that an algorithms' performance depends on the problem "characteristics".

III. Methodology

3.1 Introduction

The heuristic methods proposed by Toyoda, Senju-Toyoda (S – T) and Loulou – Michaelides (L – M) are examined in this section. The *Test problem characteristics in the library* section describes the test problems characteristics of those problems solved by each heuristic method. The *Approaches to the empirical analysis of algorithms and heuristics* section discusses general heuristic approaches to the empirical analysis of algorithms and heuristics. The final sections discuss how to analyze comparative data and how to compare heuristic performance.

3.2 Test problem characteristics in the library

Our objective is to examine heuristic performance as a function of constraint tightness and problem correlation structure. We used the Spearman 2KP problems developed and used by Hill and Reilly (2000). For each problem, the number of constraints is 2 (i.e., the 2KP), and the number of variables is 100. These test problems were created using a Spearman rank correlation induction method. This method creates values of trivariate random variables to represent the coefficients (c_j, a_{1j}, a_{2j}), and ensure the sets of values have the desired correlation structure. The objective function coefficients, c_j, are integer numbers uniformly distributed from 1 to 100. The coefficients of the first constraint, a_{1j}, are integer numbers uniformly distributed from 1 to 25 while the coefficients of the second constraint, a_{2j}, are integer numbers uniformly distributed from 1 to 40. The three correlation terms are ρ_{CA^1}, ρ_{CA^2}, and $\rho_{A^1A^2}$. The terms ρ_{CA^1} and

ρ_{CA^2} represent the correlation between objective function coefficients (c_j) and constraint coefficients (a_{1j} and a_{2j}). The term $\rho_{A^1A^2}$ represents the correlation between the two constraint coefficients. The range of correlation levels for each correlation term are set as follows:

$$\rho_{CA^1} \in \{-0.99997, -0.49999, 0, 0.49999, 0.99997\}$$

$$\rho_{CA^2} \in \{-0.99773, -0.49887, 0, 0.49887, 0.99773\}$$

$$\rho_{A^1A^2} \in \{-0.99752, -0.49876, 0, 0.49876, 0.99752\}$$

Within each set of correlation values, the largest absolute values represent the extreme correlation level. Considering each possible combination of correlation value implies 125 combinations. However, of these 125, only 45 represent positive definite correlation matrices. For correlation structure, the five levels of correlation for each correlation term are coded as {–2, –1, 0, 1, 2}.

For constraint slackness, two different constraint slackness are examined in this study. Slackness number of 1 indicates a slackness of 0.30 and a slackness number of 2 indicates a slackness of 0.70. The right-hand side coefficients (b_i) are set using the relation:

$$b_i = S_i \sum_{j \in N} a_{ij}$$

where $S_1 = 0.30$ and $S_2 = 0.70$.

Each of the four possible setting of S_1 and S_2 is referred to as a constraint slackness setting (0.30 or 0.70). We have a total of 1120 problems with 224

combinations of 45 feasible correlation structures, four-constraint slackness settings, and 5 replications each.

3.3 Approaches to the empirical analysis of algorithms and heuristics

Barr *et al*. (1995) outline a general approach for conducting empirical testing of heuristics. We follow the guidelines for the empirical testing of heuristics by Barr *et al*:

1. Define the goals of the experiment:

My goal is to conduct a rigorous computational study to isolate and examine the performance of three greedy heuristics, TOYODA, Senju – Toyoda, Loulou – Michaelides based on constraint slackness and correlation structure. The purpose of this study is to conduct a computational test, gain insight into how correlation structure and constraints slackness affect three different heuristics, and develop a new heuristic based on the results. In other words, which heuristic method yields the best solution under certain correlation and slackness conditions and how might we take advantage of this knowledge.

2. Choose measures of performance and factors to explore:

A factor is any controllable variable in an experiment that effects the outcome of the experiment. The factors, in this experiment, are the four-constraint slackness (combination of tight and loose constraints) levels and 45 feasible correlation structures.

A measure is the outcome of an experiment. All three heuristics were coded in Visual Basic for Applications within Excel and run against the problems. Each combination of constraint slackness and correlation yields a best heuristic method and by

counting the number of times each method is best for various factor combinations, we can measure which heuristic is best for each problem strucuture.

3. Design and execute the experiment:

As mentioned in Section 3.2, we use the1120 test problem set from Hill and Reilly (2000). These problems represent a full-factorial design in constraint slackness and all feasible combinations of the correlation value sets.

4. Analyze the data and draw conclusions:

Barr *et al*. suggested that there were at least three sources of variation one must recognize. These are as follows: (1) variation among algorithm performance, (2) variation due to problem parameters and (3) variation within problems. Based on Hill and Reilly (2000), (3) is negligible. We therefore focus on (1) and (2) with emphasis on (1) since we hope to gain insight into why certain heuristics do well or not so well on certain problems.

3.4 Computer coding of three heuristic methods

In order to eliminate the possible influences caused by any type of restriction, each constraint is normalized by dividing all constraint row coefficients (a_{ij}) by its right-hand side coefficient (b_i), yielding a problem with each $b_i = 1$.

For TOYODA's heuristic, we coded the original primal effective gradient method. Toyoda (1975) included origin – moving in his primal effective gradient method, but we did not use this. Let F_{ij} be the normalized constraint coefficients (a_{ij}). When cumulative quantity vector is a zero vector such as in the first iteration, the

gradient for each variable is calculated as $G_j = \frac{c_j \cdot \sqrt{2}}{(F_{1j} + F_{2j})}$. Let P_j be a vector of (F_{1j}, F_{2j}). For most iterations, the effective gradient is $G_j = \frac{c_j \cdot |P_u|}{P_j \cdot P_u}$, where P_u is cumulative total quantity vector, i.e., the vector of used resources. Variables are added according to best effective gradient until the constraint resources are fully used.

For S – T heuristic, we coded the improved method suggested by the authors. The improved method also normalizes constraint coefficients. The following vectors are introduced: A_i = vector of constraint coefficients which is normalized, $A_i = (a_{i1}, a_{i2})$. R is the sum vector of A_i. B is the vector of right – hand side values. Since RHS is normalized, $B = (1, 1)$. S is slackness vector. The following vector equations hold:

$$R = A_1 + A_2 + \cdots + A_{100}$$

$$S = R - B$$

The gradient is calculated as: $G_j = \frac{c_j}{P_j \cdot S}$. Then, since S – T is a dual approach, S – T drops variables according to best effective gradient until feasibility is achieved.

As mentioned in section 2.3.2, for the L – M heuristic, Loulou and Michaelides (1979) suggested four different methods to find the penalty vectors. We use method M1 to define the penalty vector. That is:

$$V_j = Max_{i=1,\ldots,100}\{(DA_i + a_{ij})(\sum_{k \in SC} a_{ik} - a_{ij})/(1 - DA_i - a_{ij})\}$$

where DA_i is an amount of resource consumed so far.

Notice L – M heuristic picks the worst penalty cost (V_j) between two constraints. The effective gradient is calculated as: $G_j = \frac{c_j}{V_j}$. L – M adds variables according to the effective gradient until constraints are met.

All three heuristic codes are available in Appendix A, B and C, respectively. Some representative results are shown on Table 1. All problems were successfully solved.

Table 1. Results by Three Heuristics Solutions over 1120 problems

Prob Num	Rep Num	C1 Slack	C2 Slack	CA1	CA2	A1A2	IP	TOYODA	S-T	L-M
1	1	1	1	2	2	2	1480	1468	1468	1468
2	2	1	1	2	2	2	1644	1631	1614	1584
3	3	1	1	2	2	2	1497	1454	1454	1441
4	4	1	1	2	2	2	1704	1694	1696	1603
5	5	1	1	2	2	2	1619	1597	1597	1549
6	1	1	2	2	2	2	1647	1626	1635	1611
7	2	1	2	2	2	2	1787	1772	1770	1594
8	3	1	2	2	2	2	1590	1586	1585	1552
9	4	1	2	2	2	2	1629	1627	1628	1461
10	5	1	2	2	2	2	1669	1663	1665	1574
11	1	2	1	2	2	2	1732	1722	1725	1647
:										
:										
1117	2	2	2	2	-2	-2	3574	3540	3535	3537
1118	3	2	2	2	-2	-2	3683	3623	3620	3640
1119	4	2	2	2	-2	-2	3812	3544	3435	3532
1120	5	2	2	2	-2	-2	3696	3648	3631	3646

Prob Num = Problem Number
Rep Num = number of replications
C1 Slack = slackness of first constraint (1 = 0.3, 2 = 0.7)
C2 Slack = slackness of second constraint (1 = 0.3, 2 = 0.7)
CA1 = correlation between c_j and a_{1j}
(–2 = –0.99997, –1 = –0.49999, 0 = 0, 1 = 0.49999, 2 = 0.99997)
CA2 = correlation between c_j and a_{2j}
(–2 = –0.99773, –1 = –0.49887, 0 = 0, 1 = 0.49887, 2 = 0.99773)
A1A2 = correlation between a_{1j} and a_{2j}
(–2 = –0.99752, –1 = –0.49876, 0 = 0, 1 = 0.49876, 2 = 0.99752)
IP = Integer optimal solution (or best known)
TOYODA = solution by Toyoda's heuristic
S-T = solution by Senju – Toyoda's heuristic
L-M = solution by Loulou – Michaelides' heuristic

3.5 Analysis of Results

3.5.1 Relative Error

The ultimate goal of a heuristic is to find an optimal solution. Short of this, the heuristic should come near the optimal. Since we know the true optimum or best known solution for each test problem, we can use the relative error measure, that is, the smallest relative error provides the closet solution to the optimum. Let Z_i be the value of objective function obtained by heuristic i where i = Toyoda, S-T or L-M and Z^* be the optimal or best known solution. Then the relative error:

$$RE_i = 100 \cdot (Z^* - Z_i) / Z^*$$

Based on the relative error, we choose the best method excluding ties under certain correlation and constraint slackness. Since our goal is to know why certain correlation and constraint slackness levels make specific methods perform well (i.e., the smallest relative error), we count the number of times a heuristic is the best, excluding the number of ties, by correlation structure and constraint slackness settings. This is shown Table 2.

Table 2. Relative Error of Three Heuristics

Problem No.	TOYODA	S – T	L – M	Best Method
1	0.810811	0.810811	0.810811	Tie
2	0.790754	1.824818	3.649635	TOYODA
3	2.872411	2.872411	3.740815	Tie
4	0.586854	0.469484	5.92723	S – T
5	1.358863	1.358863	4.323657	Tie
6	1.275046	0.728597	2.185792	S – T
7	0.839396	0.951315	10.80022	TOYODA
8	0.251572	0.314465	2.389937	TOYODA
:				

3.5.2 Chi-square test

We wish to know whether or not there is a best method among the three heuristics over each correlation and constraint slackness. We use a Chi-square (χ^2) test to determine whether or not any heuristic method is significantly better than the other methods.

H_0 : Methods do not differ.

H_1 : At least 2 methods differ.

To compute the chi-square test, first divide the entire range of the fitted distribution into 3 equal intervals $[a_0, a_1)$, $[a_1, a_2)$, $[a_2, a_3)$ where $[a_0\ a_1)$ represents Toyoda's heuristic, $[a_1, a_2)$ represents Senju-Toyoda's heuristic, and $[a_2, a_3)$ represents Loulou – Michaelides's heuristic. Then we have

N_j = number of times best in the interval $[a_{j-1}, a_j)$, for j =1 , 2, 3

(Note that 1 = Toyoda, 2 = S – T, 3 = L – M heuristic)

Next, we compute the expected proportion, p_j, of the N_j's that should fall in the jth interval. Since, under H_0 we assume that there is no difference in the three methods, we can define p_j= 1/3. Finally, the test statistic is

$$\chi^2 = \sum_{j=1}^{3} \frac{(N_j - np_j)^2}{np_j}$$

Since np_j is the expected number of times the best j heuristic occurs, if H_0 is true, we would expect χ^2 to be small if the fit were good. We reject H_0 if χ^2 is too large. For this test, $\alpha = 0.1$ is used, so the critical value is $\chi^2_{0.01,\text{d.f.}}$.

3.5.3 Sign test for paired samples

We use a sign test to determine whether or not one heuristic outperforms another. For a sign test, the hypothesis is as follows:

H_0: two heuristics are identically distributed.

H_1: two heuristics are statistically different.

If H_0 is true, then for any test problem either heuristic has an equal chance of being the best. Therefore, the distribution of outcomes has the Binomial distribution B(n, 0.5). Let U be the number of times the first heuristic is best. If H_0 is true, then $U \sim B(N, 0.5)$ and is approximated by a normal distribution having mean $\mu = N \times \frac{1}{2}$ and standard deviation $\sigma = \sqrt{N \times \frac{1}{2} \times \frac{1}{2}}$. To find the significance level of the result, we calculate :

$$P(X \geq U) \approx P(X > U - 0.5) = P\left(Z > \left|\frac{U - 0.5 - \mu}{\sigma}\right|\right)$$

Since this is a two-tailed test, and the normal distribution is symmetric, we use an absolute value on the critical value. We use $\alpha = 0.05$ level of significance to decide whether to fail to reject H_0 or reject H_0.

3.6 New heuristic

3.6.1 New Combined Heuristic

We will study which heuristic has the best solution under various conditions, such as different constraint slackness settings, correlation structures or a combination of both. In other words, which heuristic under what conditions becomes statistically the best heuristic. We use the condition which made one of the three heuristics the best heuristic for a new combined heuristic. That is, we first investigate the characteristics of a

problem as to whether it is dominated by constraint slackness, correlation structures, or a combination of both. Then, we choose a heuristic that is likely to produce the closest solution to the best-known solutions among the three heuristics, TOYODA, S – T, and L – M.

3.6.2 Cho Heuristic

After studying why the best heuristic works well and why the poorer heuristic does not work well in Chapter IV, we combine the merits of the three heuristics into one, new heuristic. Since three heuristics use different penalty factors as well as feasibility, for example, TOYODA and L – M start from an empty solution and maintain feasibility while S – T starts from an infeasibility region, we may predict that some other factors are influential in creating a better solution. Therefore, we develop a new heuristic that includes the favorable influential factors.

IV. Analysis of Legacy Heuristics

This chapter presents the results of an empirical study of the Senju – Toyoda (1968), Toyoda (1975), and Loulou – Michaelides (1979) heuristics as applied to the 2KP test problem set of Hill and Reilly (2000). The purpose of this study was to gain insight into heuristic performance to build new heuristics for MKPs.

4.1 Constraints slackness

The *Results* section deals with the performance of three heuristics over four different combinations of constraint slackness (tight, loose). Most of the analysis results present data as counts of best performer (excluding ties) among the three heuristics. Graphics are validated using non-parametric statistical test.

4.1.1 Results

The overall performance of the three heuristics is summarized in Table 3 and Figure 1. The data reflect the various combinations of constraint slackness coded as: 1 = 0.3 and 2 = 0.7.

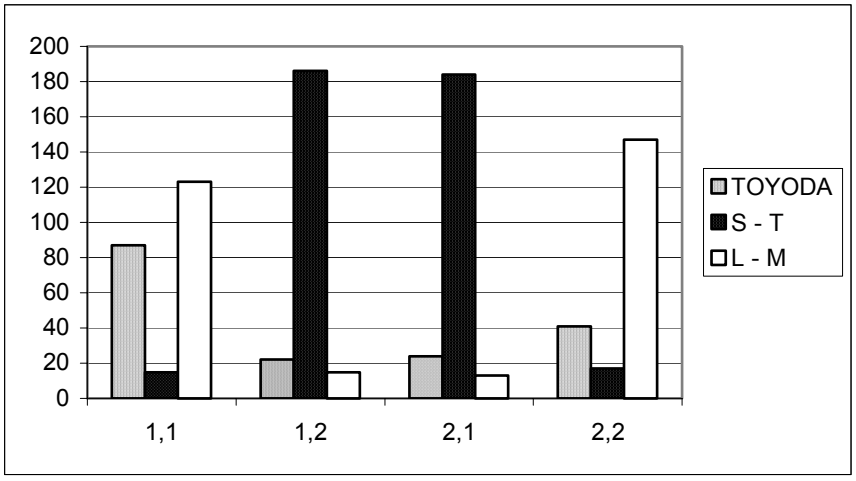

Figure 1. Counts of best of three heuristic methods under constraint slackness

Toyoda has long been the benchmark greedy heuristic for MKP problems. These results, however, suggest that S – T might be better in some cases and L – M better in other cases. The counts provided in Table 3 appear convincing; there is a difference in heuristic performance.

Table 3. The number of times best by each heuristic under constraint slackness

Constraint Slackness	Method		
	Toyoda	S - T	L - M
1,1	87	15	123
1,2	22	186	15
2,1	24	184	13
2,2	41	17	147

If there were no differences in performance of these heuristics, we would expect each row in Table 3 to have fairly equal counts. A chi-square test of

$\mathbf{H_0}$: All heuristics equal

$\mathbf{H_1}$: Heuristics differ

is summarized in Table 4. These results confirm the intuition from Table 3; for each slackness setting there is a preferred heuristic. The following sections examine each slackness setting in detail.

Table 4. Chi-square test for constraint slackness

Constraint Slackness	X^2	df	Probability	Reject Region ($\alpha = 0.1$)
1,1	80.64	2	3.08E-18	**Reject H_0**
1,2	251.9552	2	1.94E-55	**Reject H_0**
2,1	248.6968	2	9.91E-55	**Reject H_0**
2,2	140.0585	2	3.86E-31	**Reject H_0**

4.1.2 Analysis of constraint slackness of $S_1 = 0.3$ and $S_2 = 0.3$ (1,1)

Table 3 and Figure 1 indicate that TOYODA and L – M yield better solutions than S – T when constraints are tight. The question is whether TOYODA and L- M differ. A sign – test is used to examine this question. Table 5 summarized the sign test indicating there is no difference between TOYODA and L – M for $(S_1, S_2) = (0.3, 0.3)$ problems.

Table 5. Sign test, TOYODA vs. L – M under $S_1 = 0.3$ and $S_2 = 0.3$

No. of Non-Ties	No. of Ties	TOYODA Better	L – M Better
272	8	168	124
H_0	**Two heuristics are identically distributed.**		
Normal Approximation $B(272, 0.5)$	$P(X \geq 148) \approx P(Z > \frac{147.5 - 136}{8.2462}) = 0.08157$ Since this is a two-tailed test, the value $U = 168$ would be significant at a level of 0.1631		
Rejection Region ($\alpha = 0.05$)	Fail to reject **H_0**		

The challenge is understanding why TOYODA (and L – M) beats S – T for these problems. Toyoda's heuristic is a primal effective method. The penalty function creates a single effective gradient number based on two limited resources. The penalty cost was introduced as follows:

$$U_i = \frac{(P_i \cdot P_u)}{|P_u|}$$

where P_i is the vector (a_{1i}, a_{2i}) and P_u is a cumulative total resource used vector. Recall a_{1i} and a_{2i} are normalized coefficients in each constraint, i.e., $a_{ij} = \frac{F_{ij}}{b_j}$ where F_{ij} is the original coefficient in constraints. Therefore, U_i depends on the direction of P_u and has

no relationship to its magnitude. To pick the next new variable to add among non-selected variables, TOYODA calculates an effective gradient $\left(G_i = \frac{c_j}{U_i} \right)$. However, for Senju –Toyoda's heuristic, the penalty is as follows:

$$U_j = \frac{(P_j \cdot S)}{|S|}$$

where P_i is the vector (a_{1i}, a_{2i}), S is a slack variable vector, and a_{1i} and a_{2i} are equivalent to each normalized constraint multiplied by 100, i.e., $a_{ij} = \frac{F_{ij}}{b_j} \times 100$. Let R be the vector of resource costs for constraint j, such as $R = \left\{ R_j : R_j = sum\left(a_{ij}\right), i = 1,2 \right\}$. Thus $S = R - 100$.

The reason why TOYODA and L – M yield better solutions is caused by P_u. When the value S is chosen, S does not change at all in Senju-Toyoda's method, while P_u varies at each iteration in Toyoda's heuristic. Although S always provides a direction into the feasible region, S is not effective because it gives constraints equal weight and goes too deep into the feasible region. This causes more resources to remain. However, the TOYODA heuristic evaluates P_u at each iteration after selecting a new variable. This means there are less remaining resources at each constraint. Thus, Toyoda selects more variables because it uses constraint resources effectively. The Loulou–Michaelides heuristic, an extension of Toyoda's yields similar results.

4.1.3 Analysis of mixed slackness, $S_1 \neq S_2$, (1,2) or (2,1)

Recall from Figure 1 the overwhelming advantage of S – T when slackness levels are mixed. Since we normalized constraint coefficients, the coefficients, a_{ij}, represent a

percentage of the right-hand side resource. Senju and Toyoda (1968) even suggest that better solutions may be obtained when the right-hand side values differ greatly.

To understand why S – T performs well, consider problems when a (S_1, S_2) = (0.3, 0.7). Since the first constraint is tight and the second constraint is loose, and *S* is slack variable vector (s_1, s_2), s_1 should be larger than s_2.

Figure 2, adapted form Senju and Toyoda (1968), depicts the S – T heuristic approach. Let the axes represents resource average within each constraint so L_1 and L_2 represent right – hand side values for each constraint. The point R represents the resource usage of the initial, infeasible, point used in the S – T heuristic. The S – T heuristic drops variables to force the point R into the feasible region, ideally along vector *S*.

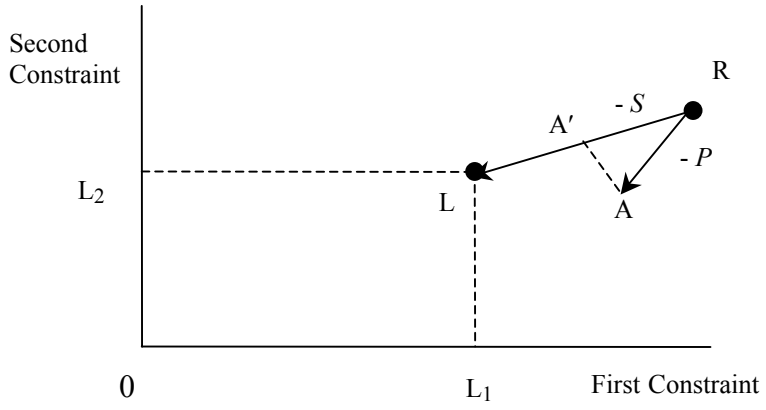

Figure 2. Effective length of withdrawal

The slope of *S* is almost horizontal with respect to first constraint axis, as shown in Figure 2. This means favoring the first constraint over the second constraint, which in the current case is the tight constraint. It is preferable to drop variables whose coefficients are smaller compared to their projected length on the vector *S*, thus favoring feasibility with respect to the tight constraint.

The direction of slack variable vector, *S*, provides the proper direction into feasible region. In other words, as S – T reaches the point of feasible region (*L*), it is using the resources of the two constraint most effectively. In contrast to the S – T dual method, the TOYODA and L – M primal methods pick variables to equalize resource usage. This approach does not provide enough emphasis towards the problem's binding (tighter) constraint. Therefore, while S – T in problems with tight constraints uses constraint resources similarly compared with TOYODA and L – M; however, with mixed constraint slackness, S – T is more effective. More effective use of the binding constraint translates into extra resources and thus more projects selected yielding better solutions.

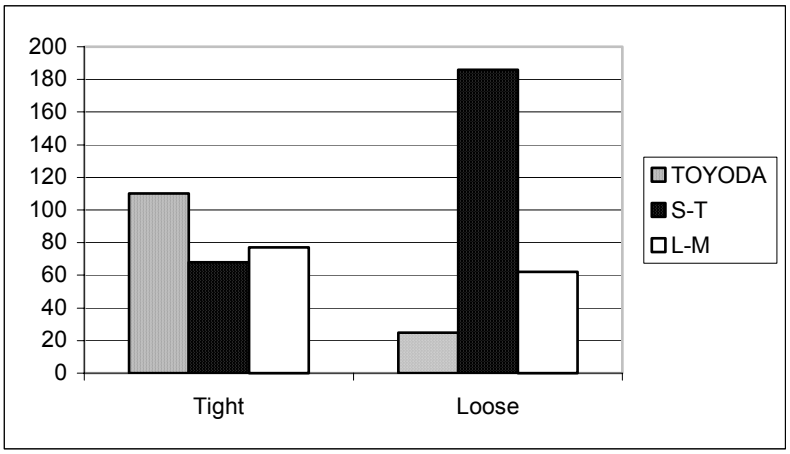

Figure 3. Performance of 3 heuristic methods in terms of best constraint resource use

The data was examined to determine which heuristic makes best use of each constraint i.e., leaves the least amount of slack in a constraint. Table 6 provides data pertaining to test problem resource usage; Figure 3 is a graph of the same data, when (S_1, S_2) = (0.3, 0.7). Table 7 and Figure 4 provide similar data for (S_1, S_2) = (0.7, 0.3). The improved direction of S – T focused on the binding constraint means more effective use of the loose constraint resource. As the data shows, constraint usage is nearly equal on the binding constraint but significantly better on the loose constraint for S – T. This equates to more variables set to one under S – T which in turn helps explain S – T's significantly better performance in terms of solution value over TOYODA and L – M.

Table 6. The number of times least slack remained in each constraint (Excluding ties)

	TOYODA	S – T	L – M
Tight Constraint (S_1 = 0.3)	110	68	77
Loose Constraint (S_2 = 0.7)	25	186	62

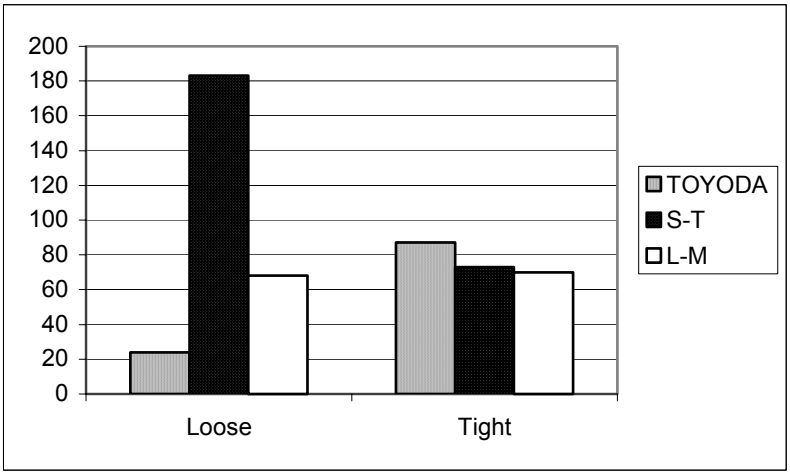

Figure 4. Performance of 3 heuristic methods in terms of best constraint resource use

Table 7. The number of times least slack remained in each constraint (Excluding ties)

	TOYODA	S – T	L – M
Loose Constraint ($S_1 = 0.7$)	24	183	68
Tight Constraint ($S_2 = 0.3$)	87	73	70

4.1.4 Analysis of constraint slackness of $S_1 = 0.7$ and $S_2 = 0.7$ (2,2*)*

In cases where both constraints are loose, TOYODA and L – M heuristics give better results than S – T. TOYODA and L – M use a similar penalty factor (V_j); furthermore, both are primal effective gradient methods because L – M is extended from TOYODA. A sign test, summarized in Table 8, shows L – M is statistically better than TOYODA on these problems.

Table 8. Sign test, TOYODA vs. L – M under $S_1 = 0.7$ and $S_2 = 0.7$

No. of Non-Ties	No. of Ties	TOYODA Better	L – M Better
266	14	113	153
H_0	**Two heuristics are identically distributed.**		
Normal Approximation $B(266, 0.5)$	$P(X \geq 113) \approx P(Z > \frac{112.5 - 133}{8.1548}) = 0.005971$ Since this is a two-tailed test, the value $U = 113$ would be significant at a level of 0.0119		
Rejection Region ($\alpha = 0.05$)	Reject **H_0**		

As described earlier, the slack vector (S) of the S – T heuristic does not provide an effective direction into the feasible region when the constraint slackness level is (2, 2). The dual direction of S yields a solution deep in the feasible region, causing more resources to remain. The second phase of S – T cannot then effectively return projects to the solution. However, TOYODA and L – M newly calculate the amount of remaining

resource to select new variables at every iteration. At the end, less resources in the constraints remain as compared to S - T. The reason for better solution of L – M under the circumstance of $S_1 = 0.7$ and $S_2 = 0.7$ results from its penalty factor. The penalty factor used is as follows:

$$V_j = Max_{i=1,\ldots,100}\{(DA_i + a_{ij})(\sum_{k \in SC} a_{ik} - a_{ij})/(1 - DA_i - a_{ij})\}$$

As we mentioned in Chapter III, when calculating the effective gradient, L – M always picks the higher value from the penalty cost between the first and second constraint. Thus, choosing a higher value of penalty cost would imply that L – M considers only the worst constraint from a penalty cost perspective.

4.1.5 Implication

The form of the effective gradient, used to either add or drop variables, is sensitive to the type of problem. In the current setting, S – T is better when slackness levels differ; TOYODA and L – M are better when both constraints are tight, and L – M is the best when both constraints are loose. This suggests analyzing problem characteristics before selecting a heuristic; a point made by Loulou and Michaelides (1979) and Hooker (1994).

4.2 Correlation

There are 45 feasible correlation structures in the test set. This section analyzes the three heuristics with respect to correlation between objective function and constraint coefficients, as well as correlation between constraint coefficients.

4.2.1 Results

Table 9 presents counts of how many times each heuristic yields the best solution (excluding ties) by problem correlation structure listed in coded form. Each correlation structure contains 20 to 40 samples (5 replications over 4 slackness settings). Figures 5 through 9 summarize the Table 9 data graphically for various "slices" of the data; each focuses on a different level of ρ_{CA}.

Table 9. The number of times best by each heuristic under correlation

Correlation	Methods		
	Toyoda	**S - T**	**L - M**
2,2,2	5	9	1
2,1,1	7	6	5
2,0,0	5	3	6
2,-1,-1	8	5	4
2,-2,-2	8	3	5
1,2,1	3	9	7
1,1,2	1	4	0
1,1,1	7	21	0
1,1,0	3	12	0
1,0,1	8	8	2
1,0,0	9	19	11
1,0,-1	3	10	7
1,-1,0	1	8	9
1,-1,-1	1	18	20
1,-1,-2	2	9	9
1,-2,-1	5	6	5
0,2,0	3	4	9
0,0,2	3	5	1
0,1,1	3	5	5
0,1,0	3	19	17
0,1,-1	2	10	7
0,0,1	9	16	1
0,0,0	6	20	2
0,-1,0	2	12	20
0,0,-1	6	22	3
0,-1,1	5	3	3
0,-1,-1	1	10	8

Correlation	Methods		
	Toyoda	S - T	L - M
0,0,-2	1	11	2
0,-2,0	1	3	10
-1,2,-1	12	3	3
-1,1,0	4	6	9
-1,1,-1	2	17	19
-1,1,-2	1	9	10
-1,0,1	3	10	3
-1,0,0	1	21	17
-1,0,-1	1	8	8
-1,-1,2	1	0	3
-1,-1,1	6	10	9
-1,-1,0	2	8	4
-1,-2,1	0	2	7
-2,2,-2	9	4	6
-2,1,-1	6	5	4
-2,0,0	3	5	9
-2,-1,1	2	3	7
-2,-2,2	0	1	0

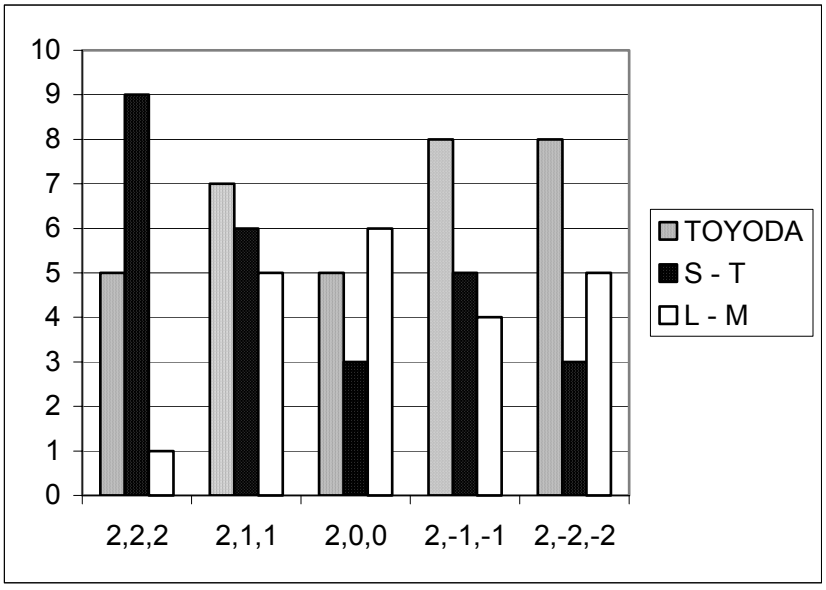

Figure 5. Performance of the three heuristic methods when ρ_{CA^1} = 2 (1)

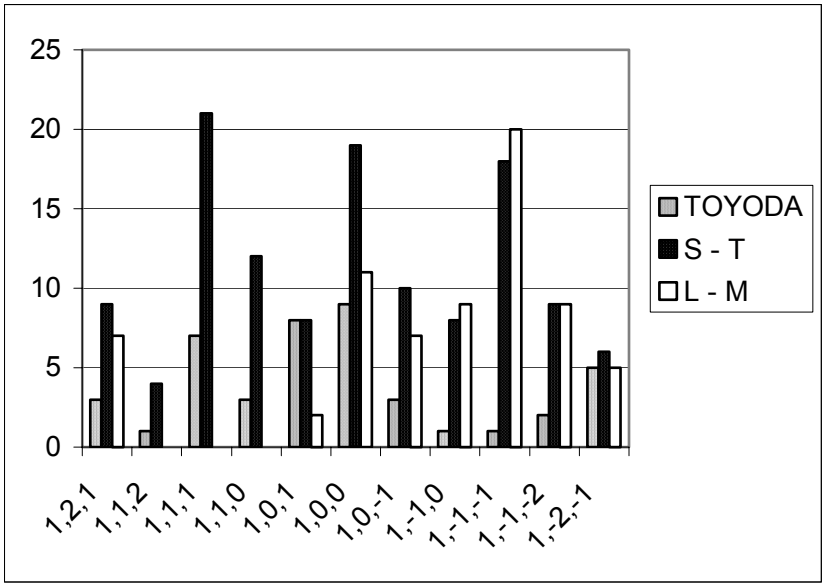

Figure 6. Performance of the three heuristic methods when ρ_{CA^1} = 1 (2)

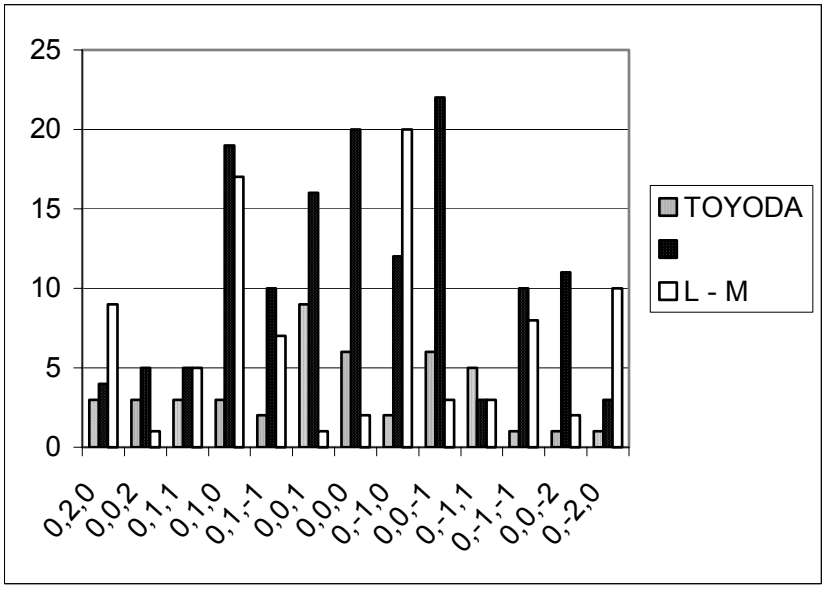

Figure 7. Performance of the three heuristic methods when ρ_{CA^1} = 0 (3)

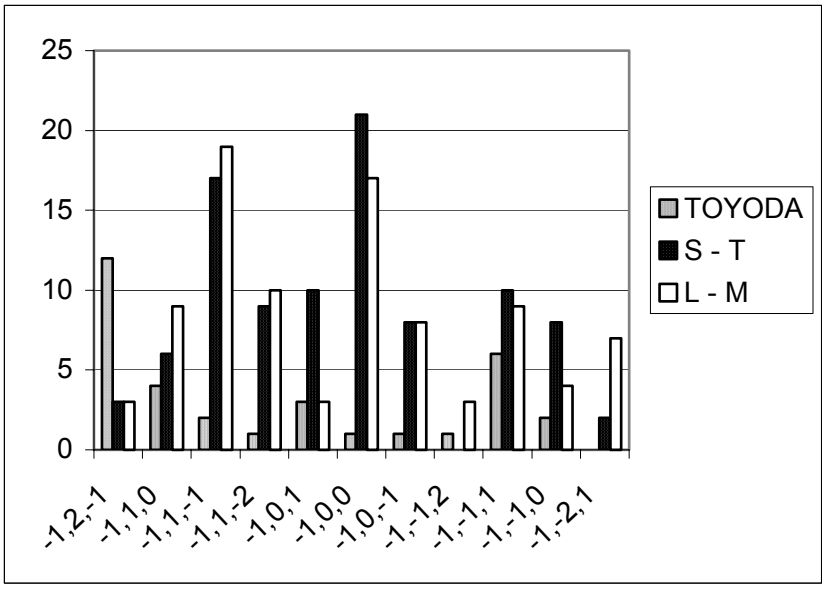

Figure 8. Performance of the three heuristic methods when ρ_{CA^1} = -1 (4)

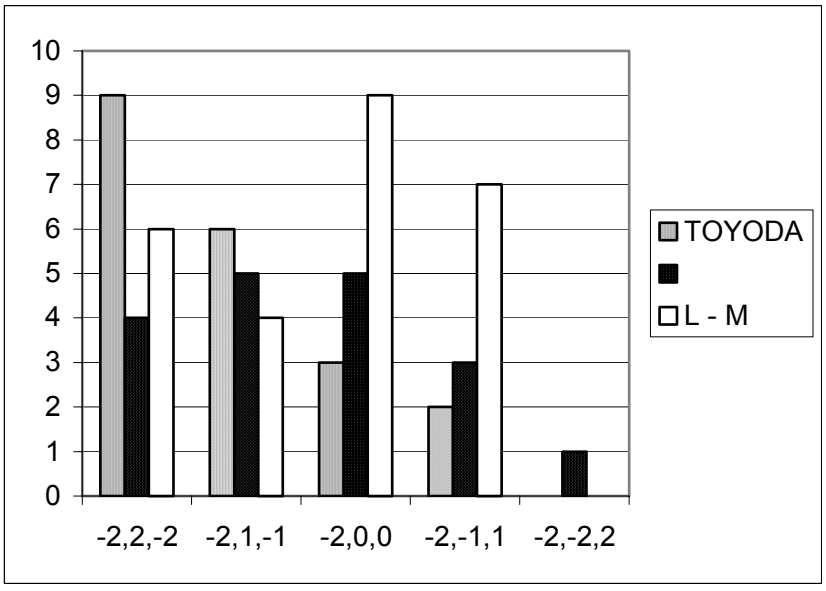

Figure 9. Performance of the three heuristic methods when ρ_{CA^1} = -2 (5)

4.2.2 Chi – square test

Table 10 summarizes the results of a Chi – Square test on each correlation structure, and for those that have it, a best performing heuristic. Clearly S – T does very well compared to TOYODA or L – M.

Table 10. Chi-square test for correlations

Correlation	X^2	df	Probability	Reject ($\alpha = 0.1$)	Best
2,2,2	6.4	2	0.040762	**Reject H_0**	**S – T**
2,1,1	0.333333	2	0.846482		
2,0,0	1	2	0.606531		
2,-1,-1	1.529412	2	0.465471		
2,-2,-2	2.375	2	0.304983		
1,2,1	2.947368	2	0.22908		
1,1,2	5.2	2	0.074274	**Reject H_0**	**S – T**
1,1,1	24.5	2	4.79E-06	**Reject H_0**	**S – T**
1,1,0	15.6	2	0.00041	**Reject H_0**	**S – T**
1,0,1	4	2	0.135335		
1,0,0	4.307692	2	0.116037		
1,0,-1	3.7	2	0.157237		
1,-1,0	6.333333	2	0.042144	**Reject H_0**	**S – T, L – M**
1,-1,-1	16.76923	2	0.000228	**Reject H_0**	**S – T, L – M**
1,-1,-2	4.9	2	0.086294	**Reject H_0**	**S – T, L – M**
1,-2,-1	0.125	2	0.939413		
0,2,0	3.875	2	0.144064		
0,0,2	2.666667	2	0.263597		
0,1,1	0.615385	2	0.735141		
0,1,0	11.69231	2	0.002891	**Reject H_0**	**S – T, L – M**
0,1,-1	5.157895	2	0.075854	**Reject H_0**	**S – T, L – M**
0,0,1	13	2	0.001503	**Reject H_0**	**S – T**
0,0,0	19.14286	2	6.97E-05	**Reject H_0**	**S – T**
0,-1,0	14.35294	2	0.000764	**Reject H_0**	**L – M**
0,0,-1	20.19355	2	4.12E-05	**Reject H_0**	**S – T**
0,-1,1	0.727273	2	0.695144		
0,-1,-1	7.052632	2	0.029413	**Reject H_0**	**S – T, L – M**
0,0,-2	13	2	0.001503	**Reject H_0**	**S – T**
0,-2,0	9.571429	2	0.008348	**Reject H_0**	**L – M**
-1,2,-1	9	2	0.011109	**Reject H_0**	**TOYODA**
-1,1,0	2	2	0.367879		
-1,1,-1	13.63158	2	0.001096	**Reject H_0**	**S – T, L – M**
-1,1,-2	7.3	2	0.025991	**Reject H_0**	**S – T, L – M**
-1,0,1	6.125	2	0.046771	**Reject H_0**	**S – T**
-1,0,0	17.23077	2	0.000181	**Reject H_0**	**S – T, L – M**
-1,0,-1	5.764706	2	0.056003	**Reject H_0**	**S – T, L – M**
-1,-1,2	3.5	2	0.173774		
-1,-1,1	1.04	2	0.594521		
-1,-1,0	4	2	0.135335		
-1,-2,1	8.666667	2	0.013124	**Reject H_0**	**L – M**
-2,2,-2	2	2	0.367879		
-2,1,-1	0.4	2	0.818731		
-2,0,0	3.294118	2	0.192616		
-2,-1,1	3.5	2	0.173774		
-2,-2,2	2	2	0.367879		

H_0 : The observed values do not differ significantly from their expected value.

We next explore why a heuristic is better under some correlation structures, and what makes heuristics perform well in general.

4.2.3 Analysis of $\rho_{CA^1} = -0.99997$, $\rho_{CA^2} = -0.99773$, and $\rho_{A^1A^2} = 0.99752$ (-2, -2, 2)

There are specific combinations of negative correlation between objective function coefficients and constraints coefficients, and positive correlation between constraint coefficients that provides the best condition for a heuristic. In constructive approaches such as the TOYODA and L – M heuristics, variables with the highest value of the effective gradient ratio c_j / V_j are selected, i.e., largest profit per unit resource consumed. For a large effective gradient, c_j should be relatively large and V_j should be relatively small with $\rho_{CA^1} = -0.99997$, $\rho_{CA^2} = -0.99773$, both constraints consume relatively less resources while profit is relatively large. Conversely, when profit is small, resource consumption will be large. Since the value of $\rho_{A^1A^2}$ is close to one, a_{1j} and a_{2j} have similar relationships in the problems. This makes the problems easy for heuristics to find best results. TOYODA and L – M found 17 optimal solutions and S – T found 19 optimal solutions out of the 20 test problems as shown below in Table 11. Similarly, conditions of weaker negative correlation between objective function coefficients and both constraints' coefficients, and stronger positive constraint coefficient correlation (-1,

Table 11. Number of times optimum found by each heuristic under correlation structure -2, -2, 2

Correlation	TOYODA	S – T	L – M
-2, -2, 2	17	19	17

-1, 2) also give good conditions for each heuristic. Under these conditions, the choice of heuristics does not matter since all do well.

4.2.4 Analysis of Toyoda's heuristic

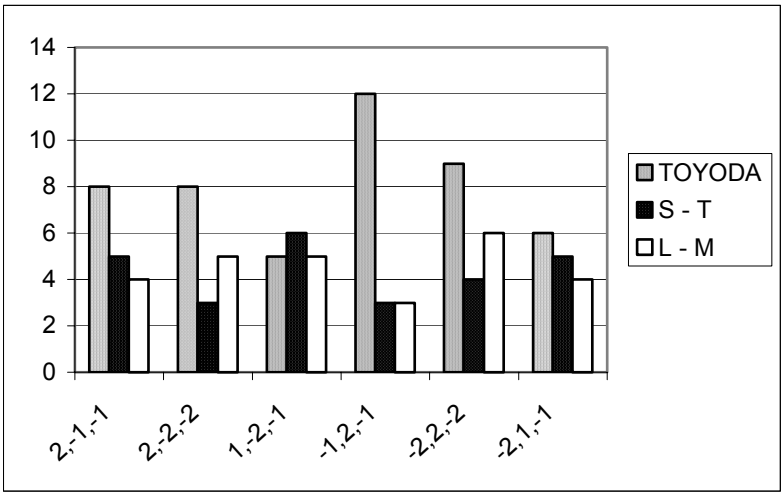

Figure 10. Better Performance of TOYODA method under various correlations

As shown in Figure 10 above, the TOYODA heuristic has better performance when ρ_{CA^1} and ρ_{CA^2} have opposite sign of correlation and $\rho_{A^1A^2}$ has negative correlation; for example $\rho_{CA^1} = -1$, $\rho_{CA^2} = 2$, and $\rho_{A^1A^2} = -1$. In other words, if the profit is large, the first constraint coefficient is relatively larger and the second constraint coefficient is relatively smaller. The TOYODA heuristic always picks the highest effective gradient. As mentioned earlier, to have a higher and effective gradient, relative profit should be large and penalty cost should be small. When constraint slackness is tight and tight or loose and loose P_u increases in almost same value at each iteration, since P_u consist of value $\sum_{i \in Selected} a_{1i}$ and $\sum_{i \in Selected} a_{2i}$ of selected variables. Since the penalty cost is P_u, multiplying by $P_i: U_i = \frac{(P_i \cdot P_u)}{|P_u|}$, for small penalty cost, relatively small values of a_{1i} plus a_{2i} are

picked. Therefore, the correlations have opposite signs and one value is larger than the other making the sum become a relatively small number. This means choosing the least resource consuming variable, meaning more variables can be selected which in turn results in better objective function value.

4.2.5 Analysis of Senju-Toyoda's heuristic

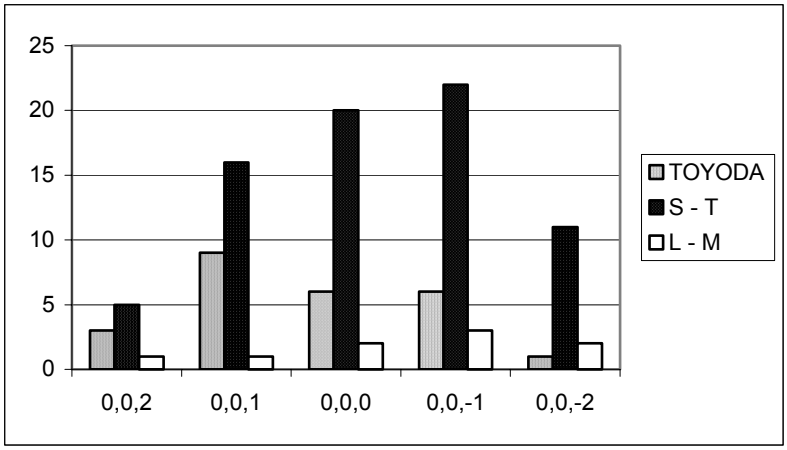

Figure 11. Better Performance of S – T method under various $\rho_{A^1A^2}$ correlations (1)

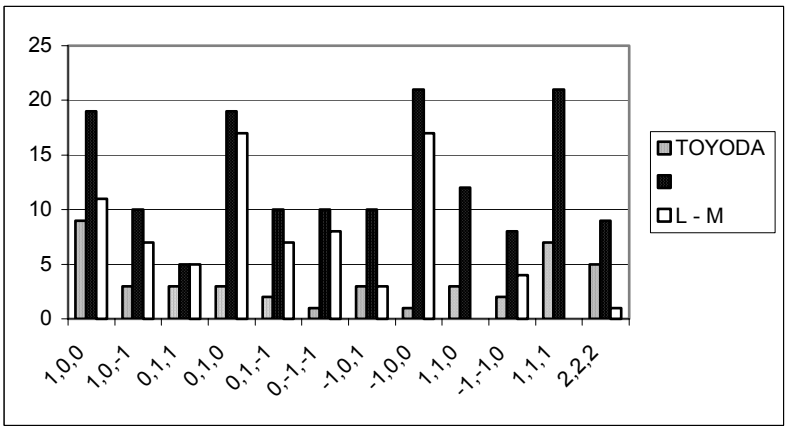

Figure 12. Better Performance of S – T method under various correlations (2)

When the ρ_{CA^1} and ρ_{CA^2} are near zero or mid – range, S – T appears to dominate. However, for each of the correlation level in Figure 11 and Figure 12 (exception is 2,2,2), 40 test problems were solved. This doubling of test problems over the other 34 correlation levels was an artifact of the Hill and Reilly (2000) experimental design. When independence or low correlation levels exist between constraint coefficients and objective function coefficients, the striking difference is caused by the S – T method's better performance on mixed constraint slackness settings; half of all problems are mixed constraints, so S – T should have better results when there are no dominant correlations.

4.2.6 Analysis of Loulou – Michaelides' heuristic

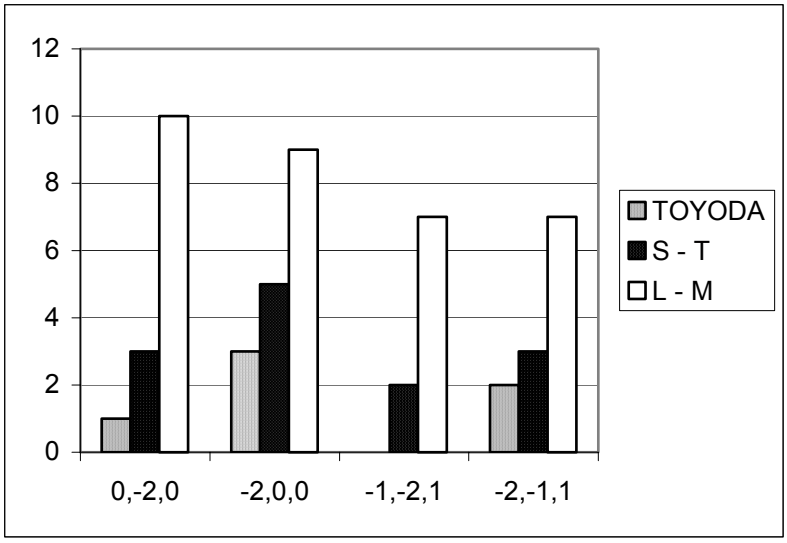

Figure 13. Better Performance of L - M method under various correlations

L – M has a tendency to have a better performance when any one constraint has strongly negative correlation ($\rho_{CA^1} = -0.99997$ or $\rho_{CA^2} = -0.99773$) with the objective function. As previously described, the best condition for a heuristic is strong negative correlation between objective function coefficients and both constraints' coefficients and

strong positive correlation between constraint coefficients. L – M heuristic always picks a penalty factor based on the worst-case constraint, i.e., it consumes more resources. Thus, inter – constraint correlation does not affect heuristic performance because L – M only considers one constraint. As shown above in Figure 13, L – M has the best performance on any combination with strong negative correlation between objective function coefficients and coefficients of one of the constraints. However, even though one of constraints has strongly negative correlation with profits, TOYODA has better results when two constraints have the opposite sign of correlation such as negative correlation and positive correlation with profits.

4.2.7 Implication

When correlation structure has strongly negative correlation between profits and constraint coefficients, primal effective methods perform better than dual methods. Especially, when opposite signs of correlation exist, TOYODA's performance is distinguishable. Since L – M is not affected by correlation of inter-constraints, L – M has better solutions when any one constraint has strongly negative correlation with profits. Except in the above condition, S – T yields more solutions close to the optimum.

V. New Heuristic Comparison

5.1 Introduction

The previous chapter examined heuristic performance as a function of problem characteristics; in particular, when is a heuristic the best choice for a problem? In this chapter, a *New Combined Heuristic* examines problem structure and runs a "best" heuristic based on the combination of constraint slackness and correlation structure used in Chapter IV. The section, *New Developed Heuristic*, provides a new heuristic where parameters have been changed based on the analysis in Chapter IV.

5.2 New Combined Heuristic

Hooker (1994) as well as Loulou and Michaelides (1979) suggest basing heuristic choice on computed problem characteristics. In Chapter IV, the best heuristic among three different heuristics, TOYODA, S – T, and L – M, was determined based on constraint slackness and correlation structure. When both constraints are tight, TOYODA and L – M, primal effective methods, are best. When constraint slackness is mixed, S – T heuristic is best. Finally, the performance of L – M is the best when both constraints are loose. Table 12 summarizes the results.

For correlation structures, TOYODA had the better performance when the sign of correlation of objective function coefficients and constraint coefficients is opposite, for example (2, –1, –1). L – M was better when any one constraint has strongly negative correlation with objective function coefficients. Table 13 summarizes the results.

Finally, a special combination between constraint slackness and correlation structure dominates the specific best heuristic. For example, when correlation is (–2, 1, –1), TOYODA is the best heuristic. However, when this correlation is combined with constraint slackness (2, 2), L – M is the best heuristic. The 20 combinations between correlation structure and constraint slackness, which dominate the specific method as the best heuristic, were found and are summarized in Table 14.

Our typology was coded and run against the problem set. This new combined heuristic code is available in Appendix D.

Table 12. Dominant Constraint Slackness and the Best Heuristic

Heuristic	Constraint Slackness 1	Constraint Slackness 2
L – M	Tight	Tight
S – T	Tight	Loose
S – T	Loose	Tight
L – M	Loose	Loose

Table 13. Dominant Correlation Structures and the Best Heuristic

Heuristic	CA1	CA2	A1A2
TOYODA	2	-1	-1
TOYODA	2	-2	-2
TOYODA	1	1	2
TOYODA	-1	2	-1
TOYODA	-2	2	-2
TOYODA	-2	1	-1
S – T	0	0	2
S – T	0	0	1
S – T	0	0	0
S – T	0	0	-1
S – T	0	0	-2
L – M	0	-2	0
L – M	-1	-2	1
L – M	-2	0	0
L – M	-2	-1	1

Table 14. Dominant Combination between Constraint Slackness and Correlation Structure, and the Best Heuristic

Heuristic	Slackness1	Slackness2	CA1	CA2	A1A2
TOYODA	1	1	2	2	2
TOYODA	1	1	2	1	1
TOYODA	1	1	1	1	1
TOYODA	1	1	1	1	0
TOYODA	1	1	1	0	1
TOYODA	1	1	1	0	0
TOYODA	1	1	1	-2	-1
TOYODA	1	1	-1	-1	2
TOYODA	2	2	1	1	1
TOYODA	2	2	1	1	0
TOYODA	2	2	0	0	1
TOYODA	2	2	0	-1	1
S – T	1	2	0	-2	0
S – T	1	2	-1	-2	1
S – T	2	1	-2	0	0
S – T	2	1	-2	-1	1
S – T	2	2	-1	0	1
L – M	1	2	-2	2	-2
L – M	1	2	-2	1	-1
L – M	2	2	-2	1	-1

5.2.1 Results under Constraint Slackness

The performance of the new combined heuristic is summarized in Table 15 and Figure 14. Table 15 and Figure 14 show that the new combined heuristic has slightly better performance than the prior best heuristic under various constraint slackness levels and has, overall, more consistent performance. Since the new combined heuristic runs the best heuristic under certain dominant constraint slackness levels and correlation structures, the new combined heuristic in Table 15 shows the number counted when the prior best heuristic and the new combined heuristic have the same value.

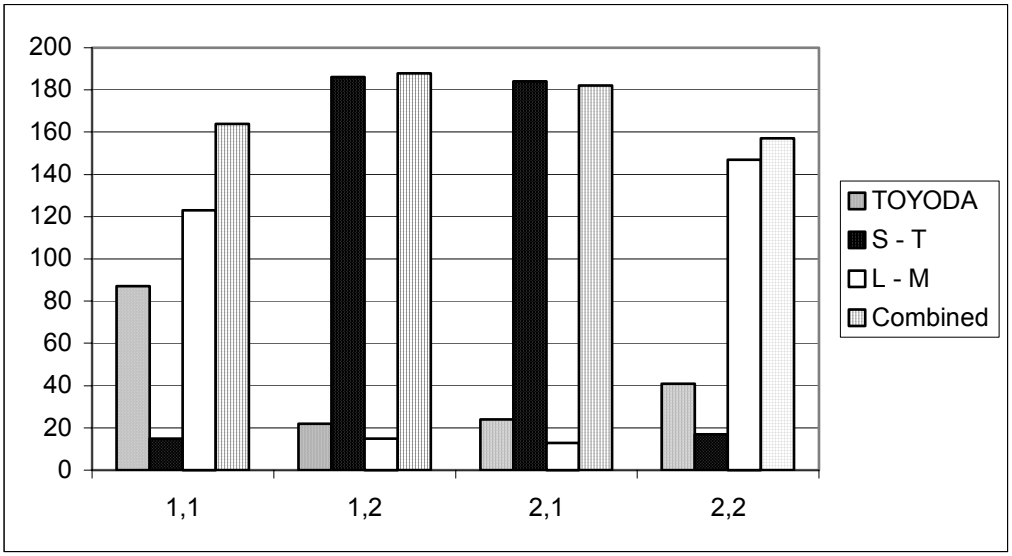

Figure 14. Performance of new combined method under various constraint slackness

Table 15. The number of times best by each heuristic under constraint slackness

Constraint Slackness	Method			
	Toyoda	S - T	L - M	New Combined
1,1	87	15	123	164
1,2	22	186	15	188
2,1	24	184	13	182
2,2	41	17	147	157

5.2.2 Results under Correlation Structure

The new approach does particularly well over correlation structure. When we count the number of same solution values of best heuristic and the new combined heuristic, the new combined heuristic performed better than the previous best heuristic. Among 45 feasible correlation structures, the new combined heuristic is best in 33

correlation structures. Figures 15 through 19 plot the data similar to Figures 5 through 9. The key trend to note is the consistent levels of performance by the combined heuristic.

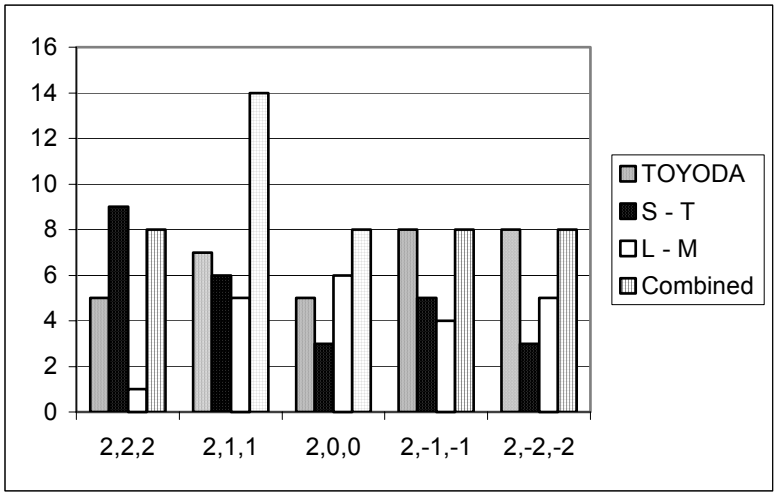

Figure 15. Performance of new combined heuristic when ρ_{CA^1} = 2 (1)

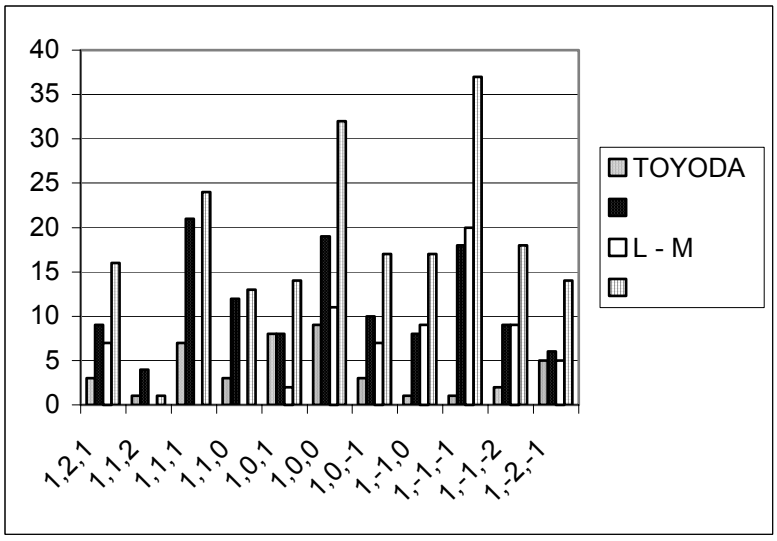

Figure 16. Performance of new combined heuristic when ρ_{CA^1} = 1 (2)

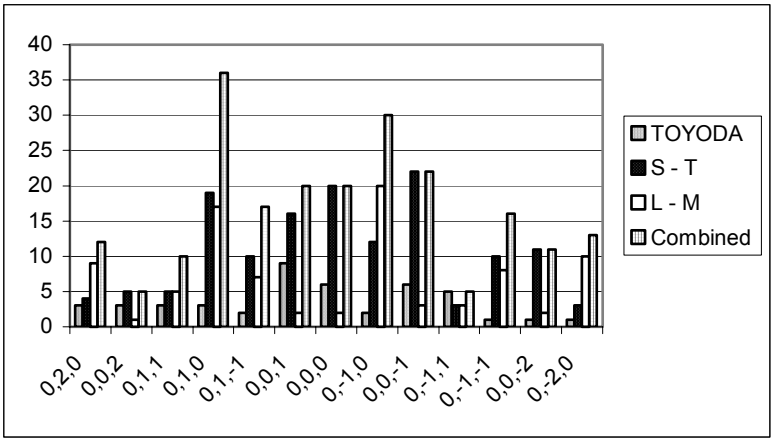

Figure 17. Performance of new combined heuristic when ρ_{CA^1} = 0 (3)

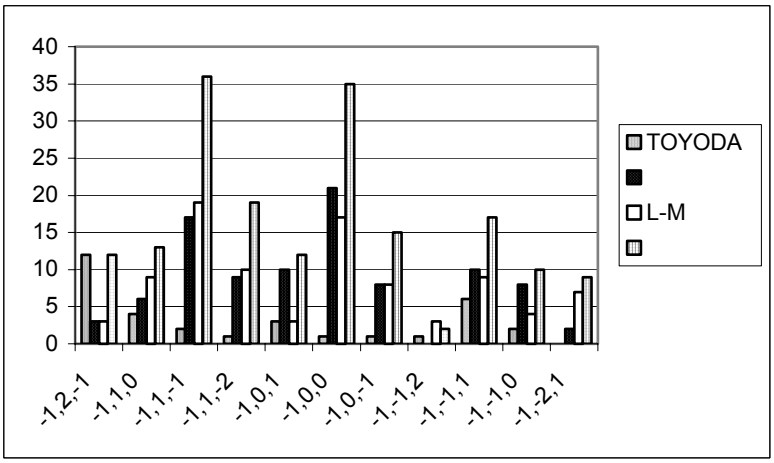

Figure 18. Performance of new combined heuristic when ρ_{CA^1} = -1 (4)

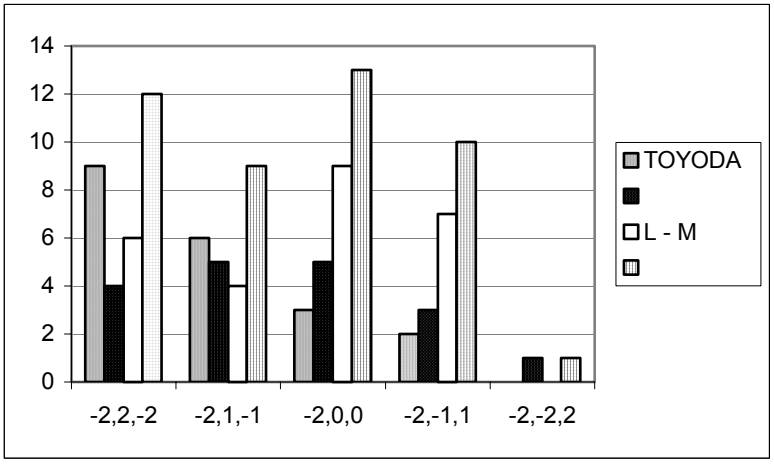

Figure 19. Performance of new combined heuristic when ρ_{CA^1} = -2 (5)

Table 16 breaks the performance numbers out by each of the 45 correlation structures in the test problem set. Clearly, the combined heuristic does well.

Table 16. The number of times best by the new combined heuristics under correlation structure

Correlation	Heuristics			
	Toyoda	S - T	L - M	New Combined
2,2,2	5	9	1	8
2,1,1	7	6	5	14
2,0,0	5	3	6	8
2,-1,-1	8	5	4	8
2,-2,-2	8	3	5	8
1,2,1	3	9	7	16
1,1,2	1	4	0	1
1,1,1	7	21	0	24
1,1,0	3	12	0	13
1,0,1	8	8	2	14
1,0,0	9	19	11	32
1,0,-1	3	10	7	17
1,-1,0	1	8	9	17
1,-1,-1	1	18	20	37
1,-1,-2	2	9	9	18
1,-2,-1	5	6	5	14
0,2,0	3	4	9	12
0,0,2	3	5	1	5
0,1,1	3	5	5	10
0,1,0	3	19	17	36
0,1,-1	2	10	7	17
0,0,1	9	16	1	20
0,0,0	6	20	2	20
0,-1,0	2	12	20	30
0,0,-1	6	22	3	22
0,-1,1	5	3	3	5
0,-1,-1	1	10	8	16
0,0,-2	1	11	2	11
0,-2,0	1	3	10	13
-1,2,-1	12	3	3	12
-1,1,0	4	6	9	13
-1,1,-1	2	17	19	36
-1,1,-2	1	9	10	19
-1,0,1	3	10	3	12
-1,0,0	1	21	17	35
-1,0,-1	1	8	8	15
-1,-1,2	1	0	3	2
-1,-1,1	6	10	9	17
-1,-1,0	2	8	4	10
-1,-2,1	0	2	7	9
-2,2,-2	9	4	6	12
-2,1,-1	6	5	4	9
-2,0,0	3	5	9	13
-2,-1,1	2	3	7	10
-2,-2,2	0	1	0	1

5.3 Cho Heuristic

As we analyzed the best heuristic in Chapter IV, the primal effective gradient methods such as TOYODA and L – M performed well when the constraint slackness levels were equal. The dual effective gradient method was the best when the constraint slackness levels were mixed. L – M, an improved version of the TOYODA heuristic, always considers the worst constraint, so it is not as affected by inter-constraint correlation. Therefore, we combined characteristics of S – T, into the L – M heuristic. The Cho heuristic is thus extended from L – M heuristic based on our knowledge of S – T from Chapter IV.

The following algorithm is a general explanation of the Cho heuristic and a comparison with Loulou and Michaelides' method. Our method differs in defining the penalty factor in step 3. We used the same symbols as Toyoda (1975): P_i = project i, i = 1, …, n; R_j = restricted resource j, j = 1, …, m; T = Set of all projects; Tu = set of projects accepted so far; T_D = set of projects not in Tu, $T - Tu$; Tc = set of candidate projects; C_j = total quantity of R_j required by the set of accepted projects, i.e., $C_i = \sum_{P_i \in Tu} a_{ij}$. Pu = cumulative total quantity vector (Toyoda, 1975). Since m = 2, $Pu = (C_1, C_2)$. The vector S_j is the surplus amount which is to subtract RHS of each constraint from sum of all row constraint coefficients for the appropriate row. Define S = slack vector as used in S – T heuristic, i.e., $S = (S_1, S_2)$ and RHS values, $B = (1, 1)$; A_i = vector of constraint coefficients, $A_i = (a_{i1}, a_{i2})$.

Step 1: Initialization.

$Tu = \varnothing$, $T_D = T$, Pu = Zero Vector

$Z = 0, \qquad X_i = 0, \qquad i = 1, \ldots, 100$

Step 2: Assign all candidate projects to *Tc*, candidate project set.

$Tc = \{P_i \mid P_i \in T_D \ and \ A_i \le 1 - Pu\}$

If $Tc = \varnothing$ STOP

Otherwise go to *Step 3*.

Step 3: Compute effective gradient for projects in *Tc* as follows:

(a) If *Pu* is a zero vector then:

$$G_i = c_i \cdot \sqrt{2} / (a_{i1} + a_{i2}) \quad \text{(Toyoda 1975)}$$

(b) Otherwise, compute;

$$U_i = \max_{j=1,2} \{(S_j \cdot a_{ij} \cdot C_j) / \sqrt{C_1^2 + C_2^2}\ \}$$

$$G_i = c_i / U_i$$

Step 4: $k = \arg\max \{ G_i \mid P_i \in Tc \text{ and feasible} \}$

Step 5: Calculate:

$Z = Z + c_{k,} \qquad X_k = 1, \qquad Tu = Tu + \{ P_k\}$

$Pu = Pu + P_k, \qquad T_D = T - \{P_k\}$

Go to *Step 2*.

Each iteration selects a new project with the largest gradient. For the penalty cost, U_i, multiplying slack vector, S_j, and constraint coefficients, a_{ij}, provides a better direction especially when two constraints greatly differ. The multiplier C_j considers the cumulative amount added so far to the *j*th constraint. For example, if the characteristic of a problem is tight and loose, S_1 should be large and S_2 should be small. Since a greedy heuristic always selects the project with the largest gradient, it selects the project whose

objective function coefficient c_j is relatively large while constraint coefficients a_{ij} is relatively small. In this case, a_{i1} is generally larger than a_{i2} because the sum of a_{i1} is much larger than the sum of a_{i2}. So, we are concerned with only a_{i1}, not a_{i2}. Multiplying S_1 and C_1 by a_{i1} make us consider only the tight constraint. This enables a better direction in the feasible region considering the tight constraint. The code is available in appendix E.

5.3.1 General Results

The overall result of the new developed method is better than each of three heuristics as shown below in Table 17. The reason is appealing since the Cho heuristic is a general purpose greedy approach not tied to problem type analysis.

Table 17. Comparison of CHO heuristic with test heuristics

Versus Heuristic	BETTER	TIE	WORSE
TOYODA	714	107	299
S – T	557	215	348
L – M	774	70	276

5.3.2 Results under various constraint slackness

The Cho heuristic really improves primal heuristic performance when constraint slackness levels are mixed. Even though this method is based on the L – M heuristic, this method is significantly better than TOYODA or L – M at constraint slackness, (1, 2), and (2, 1). As we may expect, the slack vector enables the heuristic to use resources more effectively at constraints. The results are graphed in Figure 20 and summarized in Table 18. For comparison, Cho heuristic in Table 18 shows the number counted when the prior best heuristic and the Cho heuristic have the same or better value.

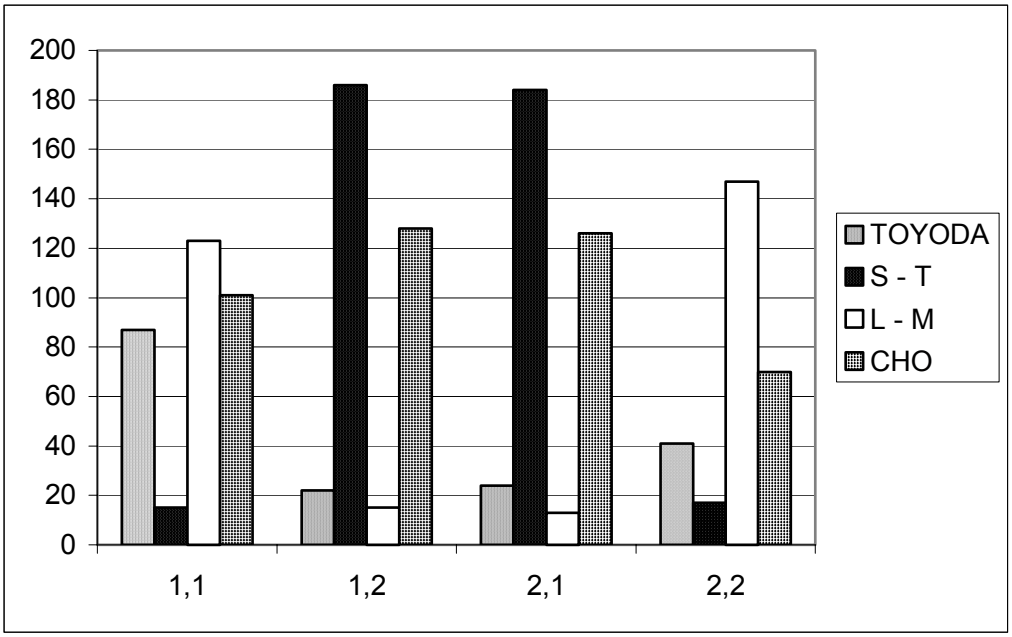

Figure 20. Performance of Cho heuristic under various constraint slackness

Table 18. The number of times best by each heuristic under constraint slackness

Constraint Slackness	Method			
	TOYODA	S - T	L - M	CHO
1,1	87	15	123	101
1,2	22	186	15	128
2,1	24	184	13	126
2,2	41	17	147	70

VI. Conclusion

6.1 Summary

The purpose of this research was to understand what makes a heuristic perform better. To accomplish this goal, 1120 problems were examined based on problem constraint slackness setting and correlation structure. The research focused on which heuristic gave the best solution under these various conditions. The methodology included two non-parametric statistic tests to prove whether or not any heuristic method was significantly better than the other methods. In the results and analyses chapter, the best heuristic was examined. We also studied why the best heuristic behaved as it did as a function of problem characteristics. Using this knowledge we examined two new heuristics: one based on problem type, a second based on a new penalty vector. These new heuristic were competitively tested against the three original heuristics and performed quite well.

6.2 Findings

Three heuristics, TOYODA; S – T; L – M, were examined under different combinations of constraint slackness and correlation structure. For a constraint slackness of tight and tight, the primal effective gradient methods such as TOYODA and L – M perform well because these methods evaluate their effective gradient at each iteration. It is very important to point out that TOYODA and L – M selects the next variable to set to 1 by recalculating the remaining resources which is the reason for yielding better solution s in comparison with S – T. For mixed constraint slackness, S – T is best because its slack vector, *S*, provides a better direction into the feasible region.

For correlation structure, Toyoda's method has better results when ρ_{CA^1} and ρ_{CA^2} have the opposite sign of correlation and $\rho_{A^1A^2}$ has negative correlation. Since TOYODA always evaluates the penalty cost at each iteration, the resource is used to keep a balance between two constraints, even though correlation has the opposite signs. The S – T method does not have any dominant correlation structure. However, when correlation does not exist among coefficients, S – T appears to yield better solution. This is mainly caused by the S – T method's better performance on mixed constraint settings. L – M has the merit that it considers one of the constraints, and is not affected by $\rho_{A^1A^2}$. Thus, when ρ_{CA^1} or ρ_{CA^2} have strongly negative correlation, L – M has better performance.

A meta-heuristic based on problem characteristics works well. Our approach employed an analysis of constraint slackness and correlation structure to choose a "best" heuristic. Test results confirmed overall consistent results.

Knowledge gained from the empirical analysis led to an improved primal greedy heuristic. Both TOYODA and L – M struggled with mixed slackness settings while S – T did not. A new penalty factor that incorporated S – T characteristics evened out the primal heuristic performance.

6.3 Recommendations

There are several areas that should be examined in future research of this topic. This thesis examined 1120 problems which are 2KP with 100 variables. One could examine more constraints, more variables, and larger test problems. In addition, one could change the correlation induction method to create more correlation settings, and

produce more constraint slackness settings. Also, various types of heuristics could be examined such as tabu search, genetic algorithms, simulated annealing, or an ant colony optimization algorithm.

Appendix A. TOYODA Heuristic Code

```
Sub TOYODA()
    Dim objcoeff(1 To 100, 1 To 1120)
    Dim rhscoeff(1 To 2, 1 To 1120)
    Dim a1coeff(1 To 100, 1 To 1120)
    Dim a2coeff(1 To 100, 1 To 1120)
    Dim objvalue(1 To 1120)

    Sheets("data").Activate
    Range("A1").Activate
    For k = 1 To 1120
        For i = 1 To 2
        rhscoeff(i, k) = ActiveCell.Offset(0, i + 3).Value
        Next i
        For i = 1 To 100
            objcoeff(i, k) = ActiveCell.Offset(1, i - 1).Value
            a1coeff(i, k) = ActiveCell.Offset(2, i - 1).Value
            a2coeff(i, k) = ActiveCell.Offset(3, i - 1).Value
            Next i
        objvalue(k) = ActiveCell.Offset(0, 3).Value
        ActiveCell.Offset(4, 0).Activate
    Next k


Dim m As Integer, n As Integer
Dim Pi(1 To 100) As Integer, R1 As Integer, R2 As Integer
Dim Ki(1 To 100) As Integer
Dim Fi1(1 To 100) As Single, Fi2(1 To 100) As Single
Dim Z As Integer
Dim Xi(1 To 100) As Integer
Dim Tu(1 To 100) As Integer
Dim Td(1 To 100) As Integer, PPi(1 To 100, 1 To 2) As Single
Dim C1 As Single, C2 As Single
Dim PPu(1 To 2) As Single
Dim B(1 To 2) As Integer
Dim Tc(1 To 100) As Integer
Dim Ui(1 To 100) As Single, Gi(1 To 100) As Single


'Step 1'

Sheets("Results").Activate
Range("C1").Activate

For k = 1 To 1120

For i = 1 To 100
    Tu(i) = 0
    Tc(i) = 0
    Td(i) = 1
    Xi(i) = 0
    Fi1(i) = a1coeff(i, k) / rhscoeff(1, k)
```

```
    Fi2(i) = a2coeff(i, k) / rhscoeff(2, k)
    Ki(i) = objcoeff(i, k)
    PPi(i, 1) = Fi1(i)
    PPi(i, 2) = Fi2(i)

Next i

PPu(1) = 0
PPu(2) = 0
R1 = rhscoeff(1, k)
R2 = rhscoeff(2, k)
m = 0
Z = 0
B(1) = 1
B(2) = 1

'Step 2'
For j = 1 To 100
  For i = 1 To 100
    If Td(i) = 1 Then
        If PPi(i, 1) <= B(1) - PPu(1) And PPi(i, 2) <= B(2) - PPu(2)
      Then
          Tc(i) = 1
        Else
          Tc(i) = 0
        End If
    End If
  Next i

'Step 3'

  n = 0
  For i = 1 To 100
    If Tc(i) = 0 Then
        n = n + 1
    End If
  Next i
   If n = 100 Then
     Exit For
   End If

'Step 4'

  If PPu(1) = 0 And PPu(2) = 0 Then
    For i = 1 To 100
        If Tc(i) = 1 Then
            Gi(i) = Ki(i) * (2 ^ 0.5) / (Fi1(i) + Fi2(i))
        End If
    Next i
  Else

    C1 = 0
    C2 = 0
    For i = 1 To 100
        If Tu(i) = 1 Then
```

```
            C1 = C1 + Fi1(i)
            C2 = C2 + Fi2(i)
        End If
    Next i

    For i = 1 To 100
        If Tc(i) = 1 Then
            If Fi1(i) * C1 + Fi2(i) * C2 = 0 Then
                Gi(i) = 100000
            End If
            Ui(i) = (Fi1(i) * C1 + Fi2(i) * C2) / ((C1 ^ 2 + C2 ^ 2) ^
      0.5)
            Gi(i) = Ki(i) / Ui(i)
        End If
    Next i
   End If

'Step 5'

  Max = 0
  For i = 1 To 100
    If Tc(i) = 1 Then
        If Gi(i) > Max Then
            Max = Gi(i)
            m = i
        End If
    End If
  Next i


'Step 6'
  Tu(m) = 1
  PPu(1) = PPu(1) + PPi(m, 1)
  PPu(2) = PPu(2) + PPi(m, 2)
  Z = Z + Ki(m)
  Td(m) = 0
  Tc(m) = 0
  Xi(m) = 1
  Range("c1").Activate

Next j


Next k

End Sub
```

Appendix B. S – T Heuristic Code

```
Sub senju()
Dim objcoeff(1 To 100, 1 To 1120)
Dim rhscoeff(1 To 2, 1 To 1120)
Dim a1coeff(1 To 100, 1 To 1120)
Dim a2coeff(1 To 100, 1 To 1120)
Dim objvalue(1 To 1120)

Sheets("data").Activate
Range("A1").Activate
For k = 1 To 1120
For i = 1 To 2
rhscoeff(i, k) = ActiveCell.Offset(0, i + 3).Value
Next i
For i = 1 To 100
objcoeff(i, k) = ActiveCell.Offset(1, i - 1).Value
a1coeff(i, k) = ActiveCell.Offset(2, i - 1).Value
a2coeff(i, k) = ActiveCell.Offset(3, i - 1).Value
Next i
objvalue(k) = ActiveCell.Offset(0, 3).Value
ActiveCell.Offset(4, 0).Activate
Next k


'run the senju heuristic'

Dim m As Integer, n As Integer, p As Integer
Dim Pi(1 To 100) As Integer, R1 As Integer, R2 As Integer
Dim Ki(1 To 100) As Integer
Dim ai1(1 To 100) As Single, ai2(1 To 100) As Single
Dim Z As Integer
Dim Xi(1 To 100) As Integer
Dim Sai1 As Single, Sai2 As Single
Dim S1 As Single, S2 As Single
Dim Ni(1 To 100) As Single
Dim Ui(1 To 100) As Single, Gi(1 To 100) As Single

'Step 1'

Sheets("Results").Activate
Range("D1").Activate


For k = 1 To 1120

    R1 = rhscoeff(1, k)
    R2 = rhscoeff(2, k)
    Sai1 = 0
    Sai2 = 0

    'assign values to the variables
    For i = 1 To 100
```

```
      Xi(i) = 1
      Pi(i) = 1
      ai1(i) = a1coeff(i, k) / rhscoeff(1, k) * 100
      ai2(i) = a2coeff(i, k) / rhscoeff(2, k) * 100
      Ki(i) = objcoeff(i, k)
      Sai1 = Sai1 + ai1(i)
      Sai2 = Sai2 + ai2(i)
   Next i

   S1 = Sai1 - 100
   S2 = Sai2 - 100

  'find Gi'
   For i = 1 To 100
      Ni(i) = S1 * ai1(i) + S2 * ai2(i)
      Gi(i) = Ki(i) / Ni(i)
   Next i

For j = 1 To 100

  'sort based on Gi
   Min = 10000
   For i = 1 To 100
     If Pi(i) = 1 Then
      If Gi(i) < Min Then
          Min = Gi(i)
          m = i
      End If
     End If
   Next i


  'drop project with the lowest gradient'
   Pi(m) = 0
   Xi(m) = 0
   S1 = S1 - ai1(m)
   S2 = S2 - ai2(m)
   Z = 0

   If S1 <= 0 And S2 <= 0 Then
      For i = 1 To 100
          Z = Z + Xi(i) * Ki(i)
      Next i

      'place Gi in order
      For i = 1 To 100
         If Xi(i) = 0 Then
           Range("z2:z102").Cells(i, 1).Value = Gi(i)
           Range("y2:y102").Cells(i, 1).Value = i
         End If
      Next i

      Range("Y2:Z101").Select
      Selection.Sort Key1:=Range("Z2"), Order1:=xlDescending,
    Header:=xlGuess, _
```

```
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

       'add back in a product

         For i = 1 To 100
                p = Range("y2:Y102").Cells(i, 1).Value
                If p > 0 Then
                 If ai1(p) <= -S1 And ai2(p) <= -S2 Then
                    Pi(p) = 1
                    Xi(p) = 1
                    S1 = S1 + ai1(p)
                    S2 = S2 + ai2(p)
                    Z = Z + Ki(p)
                 End If
                End If
           Next i
          Exit For
      End If

  Next j


Next k
End Sub
```

Appendix C. L – M Heuristic Code

```
Sub LOULOU()
Dim objcoeff(1 To 100, 1 To 1120)
Dim rhscoeff(1 To 2, 1 To 1120)
Dim a1coeff(1 To 100, 1 To 1120)
Dim a2coeff(1 To 100, 1 To 1120)
Dim objvalue(1 To 1120)
Dim objvaluehill(1 To 1120)
Dim i As Integer, j As Integer, k As Integer

Sheets("data").Activate
Range("A1").Activate
For k = 1 To 1120
For i = 1 To 2
rhscoeff(i, k) = ActiveCell.Offset(0, i + 3).Value
Next i
For i = 1 To 100
objcoeff(i, k) = ActiveCell.Offset(1, i - 1).Value
a1coeff(i, k) = ActiveCell.Offset(2, i - 1).Value
a2coeff(i, k) = ActiveCell.Offset(3, i - 1).Value
Next i
objvalue(k) = ActiveCell.Offset(0, 3).Value
objvaluehill(k) = ActiveCell.Offset(0, 13).Value
ActiveCell.Offset(4, 0).Activate

Next k

'run the L- M heuristic'

Dim m As Integer, n As Integer
Dim Pi(1 To 100) As Integer, R1 As Integer, R2 As Integer
Dim Ki(1 To 100) As Integer
Dim Fi1(1 To 100) As Single, Fi2(1 To 100) As Single
Dim Z As Integer
Dim Xi(1 To 100) As Integer
Dim Tu(1 To 100) As Integer
Dim Td(1 To 100) As Integer, PPi(1 To 100, 1 To 2) As Single
Dim DA1 As Single, DA2 As Single
Dim PPu(1 To 2) As Single
Dim B(1 To 2) As Integer
Dim Tc(1 To 100) As Integer
Dim Vi(1 To 100) As Single, Gi(1 To 100) As Single
Dim Max As Single


'Step 1'

Sheets("Results").Activate
Range("e1").Activate

For k = 1 To 1120
```

```
For i = 1 To 100
    Tu(i) = 0
    Tc(i) = 0
    Td(i) = 1
    Xi(i) = 0
    Fi1(i) = a1coeff(i, k) / rhscoeff(1, k)
    Fi2(i) = a2coeff(i, k) / rhscoeff(2, k)
    Ki(i) = objcoeff(i, k)
    PPi(i, 1) = Fi1(i)
    PPi(i, 2) = Fi2(i)
Next i

PPu(1) = 0
PPu(2) = 0
R1 = rhscoeff(1, k)
R2 = rhscoeff(2, k)
DA1 = 0
DA2 = 0
m = 0
Z = 0
B(1) = 1
B(2) = 1

'Step 2'
For j = 1 To 100
  For i = 1 To 100
    If Td(i) = 1 Then
        If PPi(i, 1) <= B(1) - PPu(1) And PPi(i, 2) <= B(2) - PPu(2)
      Then
          Tc(i) = 1
        Else
          Tc(i) = 0
        End If
    End If
  Next i

'Step 3'

  n = 0
  For i = 1 To 100
    If Tc(i) = 0 Then
        n = n + 1
    End If
  Next i
   If n = 100 Then
     Exit For
   End If

'Step 4'

  If DA1 = 1 Or DA2 = 1 Then
       Exit For
  End If
```

```
    SC1 = 0
    SC2 = 0
    For i = 1 To 100
        If Tc(i) = 1 Then
            SC1 = SC1 + Fi1(i)
            SC2 = SC2 + Fi2(i)
         End If
    Next i

   For i = 1 To 100
        If Tc(i) = 1 Then
                V1 = (DA1 + Fi1(i)) * ((SC1 - Fi1(i))) / ((1 - DA1 -
      Fi1(i) + 0.00000001))
                V2 = (DA2 + Fi2(i)) * ((SC2 - Fi2(i))) / ((1 - DA2 -
      Fi2(i) + 0.00000001))
               If V1 > V2 Then
                  Vi(i) = V1
               Else
                  Vi(i) = V2
               End If
               Gi(i) = Ki(i) / (Vi(i) + 0.0000000001)
        End If
    Next i


'Step 5'

  Max = 0
  For i = 1 To 100
    If Tc(i) = 1 Then
        If Gi(i) > Max Then
            Max = Gi(i)
            m = i
        End If
    End If
  Next i


'Step 6'
  Tu(m) = 1
  DA1 = DA1 + Fi1(m)
  DA2 = DA2 + Fi2(m)
  PPu(1) = PPu(1) + PPi(m, 1)
  PPu(2) = PPu(2) + PPi(m, 2)
  Z = Z + Ki(m)
  Td(m) = 0
  Tc(m) = 0
  Xi(m) = 1
  Range("e1").Activate

Next j

Next k

End Sub
```

Appendix D. New Combined Heuristic Code

```
Sub NewCombinedHeuristic()

Sheets("data").Activate
Range("A1").Activate
 For t = 1 To 1120
        For i = 1 To 2
        rhscoeff(i, t) = ActiveCell.Offset(0, i + 3).Value
        Next i
        For i = 1 To 100
            objcoeff(i, t) = ActiveCell.Offset(1, i - 1).Value
            a1coeff(i, t) = ActiveCell.Offset(2, i - 1).Value
            a2coeff(i, t) = ActiveCell.Offset(3, i - 1).Value
            Next i
        objvalue(t) = ActiveCell.Offset(0, 3).Value
        ActiveCell.Offset(4, 0).Activate
    Next t

Dim Sobj(1 To 1120) As Long
Dim Sai1(1 To 1120) As Long, Sai2(1 To 1120) As Long
Dim Ssqobj(1 To 1120) As Long, Ssqai1(1 To 1120) As Long
Dim Ssqai2(1 To 1120) As Long, Spdca1(1 To 1120) As Long
Dim Spdca2(1 To 1120) As Long, Spda1a2(1 To 1120) As Long
Dim slackness1(1 To 1120) As Single
Dim slackness2(1 To 1120) As Single
Dim nmCA1(1 To 1120)
Dim denCA1(1 To 1120)
Dim nmCA2(1 To 1120)
Dim denCA2(1 To 1120)
Dim nmA1A2(1 To 1120)
Dim denA1A2(1 To 1120)
Dim corrCA1(1 To 1120) As Double
Dim corrCA2(1 To 1120) As Double
Dim corrA1A2(1 To 1120) As Double

For k = 1 To 1120
Sai1(k) = 0
Sai2(k) = 0
Sobj(k) = 0
Ssqobj(k) = 0
Ssqai1(k) = 0
Ssqai2(k) = 0
Spdca1(k) = 0
Spdca2(k) = 0
Spda1a2(k) = 0

For i = 1 To 100
        Sobj(k) = Sobj(k) + objcoeff(i, k)
        Sai1(k) = Sai1(k) + a1coeff(i, k)
        Sai2(k) = Sai2(k) + a2coeff(i, k)
        Ssqobj(k) = Ssqobj(k) + (objcoeff(i, k)) ^ 2
        Ssqai1(k) = Ssqai1(k) + (a1coeff(i, k)) ^ 2
        Ssqai2(k) = Ssqai2(k) + (a2coeff(i, k)) ^ 2
```

```
        Spdca1(k) = Spdca1(k) + (objcoeff(i, k) * a1coeff(i, k))
        Spdca2(k) = Spdca2(k) + (objcoeff(i, k) * a2coeff(i, k))
        Spda1a2(k) = Spda1a2(k) + (a1coeff(i, k) * a2coeff(i, k))
Next i
Next k

For k = 1 To 1120

slackness1(k) = rhscoeff(1, k) / Sai1(k)
slackness2(k) = rhscoeff(2, k) / Sai2(k)
nmCA1(k) = 100 * Spdca1(k) - Sobj(k) * Sai1(k)
denCA1(k) = ((100 * Ssqobj(k) - ((Sobj(k)) ^ 2)) ^ 0.5) * ((100 *
      Ssqai1(k) - ((Sai1(k)) ^ 2)) ^ 0.5)
nmCA2(k) = 100 * Spdca2(k) - Sobj(k) * Sai2(k)
denCA2(k) = ((100 * Ssqobj(k) - ((Sobj(k)) ^ 2)) ^ 0.5) * ((100 *
      Ssqai2(k) - ((Sai2(k)) ^ 2)) ^ 0.5)
nmA1A2(k) = 100 * Spda1a2(k) - Sai1(k) * Sai2(k)
denA1A2(k) = ((100 * Ssqai1(k) - ((Sai1(k)) ^ 2)) ^ 0.5) * ((100 *
      Ssqai2(k) - ((Sai2(k)) ^ 2)) ^ 0.5)
corrCA1(k) = nmCA1(k) / denCA1(k)
corrCA2(k) = nmCA2(k) / denCA2(k)
corrA1A2(k) = nmA1A2(k) / denA1A2(k)
Next k


For k = 1 To 1120

If corrCA1(k) > 0.75 And corrCA2(k) < -0.25 And corrCA2(k) > -0.75 And
      corrA1A2(k) < -0.25 And corrA1A2(k) > -0.75 Then
    Call TOYODA(k)
ElseIf corrCA1(k) > 0.75 And corrCA2(k) < -0.75 And corrA1A2(k) < -0.75
      Then
    Call TOYODA(k)
ElseIf corrCA1(k) < -0.25 And corrCA1(k) > -0.75 And corrCA2(k) > 0.75
      And corrA1A2(k) < -0.25 And corrA1A2(k) > -0.75 Then
    Call TOYODA(k)
ElseIf corrCA1(k) < -0.75 And corrCA2(k) > 0.75 And corrA1A2(k) < -0.75
      Then
    If slackness1(k) < 0.45 And slackness2(k) > 0.65 Then
       Call LOULOU(k)
    Else
       Call TOYODA(k)
    End If
ElseIf corrCA1(k) < -0.75 And corrCA2(k) < 0.75 And corrCA2(k) > 0.25
      And corrA1A2(k) < -0.25 And corrA1A2(k) > -0.75 Then
    If slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
       Call LOULOU(k)
    ElseIf slackness1(k) < 0.45 And slackness2(k) > 0.65 Then
       Call LOULOU(k)
    Else
       Call TOYODA(k)
    End If
ElseIf corrCA1(k) < 0.75 And corrCA1(k) > 0.25 And corrCA2(k) > 0.25
      And corrCA2(k) < 0.75 And corrA1A2(k) > 0.75 Then
    Call TOYODA(k)
```

```
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) > -0.25
      And corrCA2(k) < 0.25 And corrA1A2(k) < -0.75 Then
    Call senju(k)
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) > -0.25
      And corrCA2(k) < 0.25 And corrA1A2(k) > -0.75 And corrA1A2(k) < -
      0.25 Then
    Call senju(k)
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) > -0.25
      And corrCA2(k) < 0.25 And corrA1A2(k) > -0.25 And corrA1A2(k) <
      0.25 Then
    Call senju(k)
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) > -0.25
      And corrCA2(k) < 0.25 And corrA1A2(k) > 0.25 And corrA1A2(k) <
      0.75 Then
    If slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
       Call TOYODA(k)
    Else
       Call senju(k)
    End If
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) > -0.25
      And corrCA2(k) < 0.25 And corrA1A2(k) > 0.75 Then
    Call senju(k)
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) < -0.75
      And corrA1A2(k) > -0.25 And corrA1A2(k) < 0.25 Then
    If slackness1(k) < 0.45 And slackness2(k) > 0.65 Then
       Call senju(k)
    Else
       Call LOULOU(k)
    End If
ElseIf corrCA1(k) < -0.75 And corrCA2(k) > -0.25 And corrCA2(k) < 0.25
      And corrA1A2(k) > -0.25 And corrA1A2(k) < 0.25 Then
    If slackness1(k) > 0.65 And slackness2(k) < 0.45 Then
       Call senju(k)
    Else
       Call LOULOU(k)
    End If
ElseIf corrCA1(k) < -0.25 And corrCA1(k) > -0.75 And corrCA2(k) < -0.75
      And corrA1A2(k) > 0.25 And corrA1A2(k) < 0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) > 0.65 Then
       Call senju(k)
    Else
        Call LOULOU(k)
    End If
ElseIf corrCA1(k) < -0.75 And corrCA2(k) < -0.25 And corrCA2(k) > -0.75
      And corrA1A2(k) > 0.25 And corrA1A2(k) < 0.75 Then
    If slackness1(k) > 0.65 And slackness2(k) < 0.45 Then
        Call senju(k)
    Else
        Call LOULOU(k)
    End If
ElseIf corrCA1(k) > 0.75 And corrCA2(k) > 0.25 And corrCA2(k) < 0.75
      And corrA1A2(k) > 0.25 And corrA1A2(k) < 0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
```

```
        Call LOULOU(k)
    Else
        Call senju(k)
    End If
ElseIf corrCA1(k) > 0.25 And corrCA1(k) < 0.75 And corrCA2(k) > 0.25
      And corrCA2(k) < 0.75 And corrA1A2(k) > 0.25 And corrA1A2(k) <
      0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call TOYODA(k)
    Else
        Call senju(k)
    End If
ElseIf corrCA1(k) > 0.25 And corrCA1(k) < 0.75 And corrCA2(k) > 0.25
      And corrCA2(k) < 0.75 And corrA1A2(k) > -0.25 And corrA1A2(k) <
      0.25 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call TOYODA(k)
    Else
        Call senju(k)
    End If
ElseIf corrCA1(k) > 0.25 And corrCA1(k) < 0.75 And corrCA2(k) < 0.25
      And corrCA2(k) > -0.25 And corrA1A2(k) > 0.25 And corrA1A2(k) <
      0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call LOULOU(k)
    Else
        Call senju(k)
    End If
ElseIf corrCA1(k) > 0.25 And corrCA1(k) < 0.75 And corrCA2(k) < 0.25
      And corrCA2(k) > -0.25 And corrA1A2(k) > -0.25 And corrA1A2(k) <
      0.25 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call LOULOU(k)
    Else
        Call senju(k)
    End If
ElseIf corrCA1(k) > 0.25 And corrCA1(k) < 0.75 And corrCA2(k) < -0.75
      And corrA1A2(k) < -0.25 And corrA1A2(k) > -0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call LOULOU(k)
    Else
        Call senju(k)
    End If
```

```
ElseIf corrCA1(k) > -0.25 And corrCA1(k) < 0.25 And corrCA2(k) < -0.25
      And corrCA2(k) > -0.75 And corrA1A2(k) > 0.25 And corrA1A2(k) <
      0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call LOULOU(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call TOYODA(k)
    Else
        Call senju(k)
    End If
ElseIf corrCA1(k) < -0.25 And corrCA1(k) > -0.75 And corrCA2(k) < 0.25
      And corrCA2(k) > -0.25 And corrA1A2(k) > 0.25 And corrA1A2(k) <
      0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call LOULOU(k)
    Else
        Call senju(k)
    End If

ElseIf corrCA1(k) < -0.25 And corrCA1(k) > -0.75 And corrCA2(k) < -0.25
      And corrCA2(k) > -0.75 And corrA1A2(k) > 0.75 Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call LOULOU(k)
    Else
        Call senju(k)
    End If

ElseIf corrCA1(k) > 0.75 And corrCA2(k) > 0.75 And corrA1A2(k) > 0.75
      Then
    If slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
        Call TOYODA(k)
    ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
        Call LOULOU(k)
    Else
        Call senju(k)
    End If


ElseIf slackness1(k) < 0.45 And slackness2(k) < 0.45 Then
   Call LOULOU(k)
ElseIf slackness1(k) > 0.65 And slackness2(k) > 0.65 Then
   Call LOULOU(k)
Else
  Call senju(k)

End If

Next k

End Sub
```

Appendix E. Cho Heuristic Code

```
Sub ChoHeuristic()

    Dim objcoeff(1 To 100, 1 To 1120)
    Dim rhscoeff(1 To 2, 1 To 1120)
    Dim a1coeff(1 To 100, 1 To 1120)
    Dim a2coeff(1 To 100, 1 To 1120)
    Dim objvalue(1 To 1120)

    Sheets("data").Activate
    Range("A1").Activate
    For k = 1 To 1120
        For i = 1 To 2
        rhscoeff(i, k) = ActiveCell.Offset(0, i + 3).Value
        Next i
        For i = 1 To 100
            objcoeff(i, k) = ActiveCell.Offset(1, i - 1).Value
            a1coeff(i, k) = ActiveCell.Offset(2, i - 1).Value
            a2coeff(i, k) = ActiveCell.Offset(3, i - 1).Value
            Next i
        objvalue(k) = ActiveCell.Offset(0, 3).Value
        ActiveCell.Offset(4, 0).Activate
    Next k


'run Cho heuristic'

Dim m As Integer, n As Integer
Dim Pi(1 To 100) As Integer, R1 As Integer, R2 As Integer
Dim Ki(1 To 100) As Integer
Dim Fi1(1 To 100) As Single, Fi2(1 To 100) As Single
Dim Z As Integer
Dim Xi(1 To 100) As Integer
Dim Tu(1 To 100) As Integer
Dim Td(1 To 100) As Integer, PPi(1 To 100, 1 To 2) As Single
Dim C1 As Single, C2 As Single
Dim PPu(1 To 2) As Single
Dim B(1 To 2) As Integer
Dim Tc(1 To 100) As Integer
Dim Ui(1 To 100) As Single, Gi(1 To 100) As Single
Dim U1(1 To 100) As Single, U2(1 To 100) As Single


'Step 1'

Sheets("Results").Activate
Range("I1").Activate

For k = 1 To 1120

For i = 1 To 100
    Tu(i) = 0
```

```
    Tc(i) = 0
    Td(i) = 1
    Xi(i) = 0
    Fi1(i) = a1coeff(i, k) / rhscoeff(1, k)
    Fi2(i) = a2coeff(i, k) / rhscoeff(2, k)
    Ki(i) = objcoeff(i, k)
    PPi(i, 1) = Fi1(i)
    PPi(i, 2) = Fi2(i)

Next i

PPu(1) = 0
PPu(2) = 0
R1 = rhscoeff(1, k)
R2 = rhscoeff(2, k)
m = 0
Z = 0
B(1) = 1
B(2) = 1

'Step 2'
For j = 1 To 100
  For i = 1 To 100
    If Td(i) = 1 Then
        If PPi(i, 1) <= B(1) - PPu(1) And PPi(i, 2) <= B(2) - PPu(2)
      Then
          Tc(i) = 1
        Else
          Tc(i) = 0
        End If
    End If
  Next i

'Step 3'

  n = 0
  For i = 1 To 100
    If Tc(i) = 0 Then
        n = n + 1
    End If
  Next i
   If n = 100 Then
     Exit For
   End If

  Sai1 = 0
  Sai2 = 0

     For i = 1 To 100
            Sai1 = Sai1 + Fi1(i)
            Sai2 = Sai2 + Fi2(i)
    Next i
    'Step 4'
    S1 = Sai1 - 1
    S2 = Sai2 - 1
```

```
  If PPu(1) = 0 And PPu(2) = 0 Then
    For i = 1 To 100
        If Tc(i) = 1 Then
            Gi(i) = Ki(i) * (2 ^ 0.5) / (Fi1(i) + Fi2(i))
        End If
    Next i
  Else

    C1 = 0
    C2 = 0
    For i = 1 To 100
        If Tu(i) = 1 Then
            C1 = C1 + Fi1(i)
            C2 = C2 + Fi2(i)
        End If

    Next i

    For i = 1 To 100
        If Tc(i) = 1 Then
            If Fi1(i) * C1 + Fi2(i) * C2 = 0 Then
                Gi(i) = 100000
            End If
            U1(i) = (S1 * Fi1(i) * C1) / ((C1 ^ 2 + C2 ^ 2) ^ 0.5)
            U2(i) = (S2 * Fi2(i) * C2) / ((C1 ^ 2 + C2 ^ 2) ^ 0.5)

            If U1(i) > U2(i) Then
               Ui(i) = U1(i)
            Else
               Ui(i) = U2(i)
            End If

            Gi(i) = Ki(i) / Ui(i)
        End If
    Next i
   End If

'Step 5'

  Max = 0
  For i = 1 To 100
    If Tc(i) = 1 Then
        If Gi(i) > Max Then
            Max = Gi(i)
            m = i
        End If
    End If
  Next i


'Step 6'
  Tu(m) = 1
  PPu(1) = PPu(1) + PPi(m, 1)
  PPu(2) = PPu(2) + PPi(m, 2)
```

```
    Z = Z + Ki(m)
    Td(m) = 0
    Tc(m) = 0
    Xi(m) = 1
    Range("I1").Activate

  Next j

  Next k

  End Sub
```

Bibliography

Barr. R., B. Golden, J. Kelly, M. Resende, and W. Stewart Jr. "Designing and reporting on computational experiments with heuristic methods," *Journal of Heuristic*, 1(1): 9-32 (1995).

Fox G. and C. Nachtsheim. "An analysis of six greedy selection rules on a class of zero-one integer programming models," *Naval Research Logistics*, 37(2): 299-307 (1990).

Fox G. and G. Scudder. "A Heuristic with Tie Breaking for Certain 0 – 1 Integer Programming Models," *Naval Research Logistics*, 32(4): 613-623 (1985).

Glover, F. "Surrogate Constraint," *Operations. Research*, 16(4): 741-749 (1968).

Glover, F. "Surrogate Constraint Duality in Mathematical Programming," *Operations. Research*, 23(3): 434-451 (1975).

Glover, F. "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, 8(1): 156-166 (1977).

Godfrey, M.G., E.M. Roebuck, and A.J. Sherlock. *Concise Statistics*. Edward Arnold, 1988.

Hill, R. R. and C. Reilly. "The Effects of Coefficient Correlation Structure," *Management Science*, 46(2) (2000).

Hiller, F. S. "Efficient Heuristic Procedures for Integer Linear Programming with an Interior," *Operations Research*, 17(2): 600-637 (1969).

Hooker, J.N. "An empirical science of algorithms," *Operations Research*, 42(2): 201-212 (1994).

Kochenberger, G., McCarl, B., and Wyman, F. "A Heuristic for General Integer Programming," *Decision Sciences*, 5(1): 36-44 (1974).

Loulou, R., and E. Michaelides. "New greedy heuristics for the multidimensional 0-1 knapsack problem," *Operations Research*, 27(6): 1101- 1114 (1979).

Martello, S. and P. Toth. "A New Algorithm for the 0 – 1 Knapsack Problem," *Management Science*, 34(5): 633-644 (1988).

Martello, S. and P. Toth. "Upper bounds and algorithms for hard 0 – 1 knapsack problems," *Operations Research*, 45(5): 768-777 (1997).

Nemhauser G. and Wolsey L. *Integer and Combinatorial Optimization*. New York: John Wiley & Sons, Inc, 1988.

Osorio, M., Glover, F and Hammer. P "Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions." *http://hces.bus.olemiss.edu/reports/hces0800.pdf*, 2002.

Pirkul, H. "A Heuristic Solution Procedure for the multiconstraint zero-one knapsack problem," *Naval Research Logistics*, 34(2): 161-172 (1987).

Senju, S. and Y. Toyoda. "An approach to linear programming with 0-1 variables," *Management Science.*, 15(4): B196- B207 (1968).

Toyoda Y. "A simplified algorithm for obtaining approximate solutions to zero-one programming problems," *Management Science.,* 21(12): 1417-1427 (1975).

Zanakis. S. H. "Heuristic 0-1 linear programming: An experimental comparison of three methods," *Management Science,* 24: 91-104 (1977).

Vita

Captain Yong Kun Cho was born in Seoul, Republic of Korea. He graduated from Nam-Kang High School in 1989. He then entered the Korea Military Academy. He received the Bachelor of Science degree in Civil Engineering in March 1993.

Upon graduation, he received the commission of 2nd Lieutenant, Army Infantry Officer. He was assigned to the 7th Division as a platoon leader. He led forty combat ready soldiers providing security against North Korean troops in the DMZ. Then he was assigned to ROK-US Combined Force Command, Seoul, Korea as Intelligence and Operation Officer. There, he conducted coordination between U.S command and ROKA providing cooperative operation for effective communication between U.S. and ROKA. He then became a company commander in Reconnaissance Battalion, 11th Infantry Division. He commanded one hundred combat ready soldiers including three officers. He conducted many company field trainings and counter-terrorism operations. He entered the Graduate School of Engineering, Air Force Institute of Technology in August 2000. He was inducted into a Tau Beta Pi (TBP-The National Engineering Honor Society) while in AFIT. Upon graduation, he will continue in the Ph.D. program at AFIT.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
08-03-2002	**Master's Thesis**	Jun 2001 – Mar 2002

4. TITLE AND SUBTITLE

EMPIRICAL ANALYSIS OF VARIOUS MULTI - DIMENSIONAL KNAPSACK HEURISTICS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Yong Kun Cho, Captain, Republic of Korea Army

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/02-04

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Since the multidimensional knapsack problems are NP-hard problems, the exact solutions of knapsack problems often need excessive computing time and storage space. Thus, heuristic approaches are more practical for multidimensional knapsack problems as problems get large. This thesis presents the results of an empirical study of the performance of heuristic solution procedures based on the coefficients correlation structures and constraint slackness settings. In this thesis, the three representative greedy heuristics, Toyoda, Senju and Toyoda, and Loulou and Michaelides' methods, are studied. The purpose of this thesis is to explore which heuristic of the three representative greedy heuristics perform best under certain combination of conditions between constraint slackness and correlation structures. This thesis examines three heuristics over 1120 problems which are all 2KPs with 100 variables created by four constraint slackness settings and 45 feasible correlation structures. Then we analyze why the best heuristic behaves as it does as a function of problem characteristic. The last chapter presents two new heuristics using knowledge gained in the study. When these new heuristics are competitively tested against the three original heuristics, the results show their better performances.

15. SUBJECT TERMS

Multi-Dimensional Knapsack Problem, Heuristic, Computational Testing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Raymond R. Hill, Lt.Col, USAF (ENS)
U	U	U	UU	92	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4323; e-mail: Raymond.Hill@afit.edu