THE STOCHASTIC KNAPSACK PROBLEM WITH DISCRETE AND CONTINUOUS RANDOM CAPACITY

INTRODUCTION

Since the advent of the industrial revolution, the world has seen a remarkable growth in the size and complexity of organization. However, it becomes more and more difficult to allocate the available resources to various activities in a way that is the most effective (i.e., maximize profits and/or efficiency, or minimize cost and/or risk) for the organization. The general allocation problems are as follows.

1. Production planning: Decide what goods to produce and how much of each.

2. Production scheduling: Decide which jobs should go on which machines in what order.

3. Transportation arrangements: Decide how to ship goods to customers at the least cost.

4. Assignment problems: Assign people to jobs, work to machines or contracts to bidders.

5. Investment planning: Select the best projects on a limited budget.

The above problems can be expressed as mathematical models and solved by various techniques. Linear programming (LP) uses a mathematical linear model to describe the problem of concern. All the mathematical functions in this model are required to be linear functions. The decision variables are continuous. However, in many practical problems, the decision variables actually make sense only if they have integer values. For example, it is often necessary to assign people, machines and

vehicles to activities in integer quantities. If required integer values are the only way in which a problem deviates from a linear programming formulation, the method to solve such problems is then called an integer linear programming (ILP) problem.

A common ILP is used to decide which subset of *n* items or projects should be selected such that the total profit is maximized without exceeding a given capacity or budget is called the knapsack problem (KP). The knapsack problem has been extensively studied in operations research (Martello and Toth, 1990). The knapsack problem has various industrial applications such as resource allocation problem, capital budgeting problem, project selection problem, cargo loading problem, cutting stock problem, and many others.

The knapsack problem of this type is deterministic because all parameters are known with certainty. However, in realistic situations, these parameters may not be known with certainty. Therefore, they have to be considered as random variables having a certain distribution. Thus, this type of problem is called the stochastic knapsack problem.

There are few researchers concerning with solving the stochastic knapsack problem, which are usually concentrated with special cases, e.g., Parks and Steinberg (1979), Sniedovich (1980) and Henig (1990) studied static stochastic knapsack problem when benefit is uncertain. Kleywegt and Papastavrou (1998) studied dynamic stochastic knapsack problem when benefit is uncertain. Chiu *et al*. (1999), Kleywegt *et al*. (1996), Slyke and Young (2000) and Kleywegt and Papastavrou (2001) studied dynamic stochastic knapsack problem when benefit and weight are uncertain.

All of the above works concern with studying uncertainty in weight parameters and/or benefit parameters. However, studying uncertainty in capacity is also important. For example, in capital budgeting problem, sometimes, the budget is uncertainty (e.g., there are *n* alternatives of budget with the probability for each alternative or the budget is random with known probability distribution). In resource

allocation problem, sometimes, the available resource is uncertainty (e.g., the available resource is random with known probability distribution). In cargo loading problem, when the truck has many customers to load the items. The available space is unknown before the truck arrived. Hence, the remaining capacity of the truck is uncertainty.

There are two types of uncertainty. One possibility is to estimate the probability of occurrence of each state of nature (i.e., there are the previous information). Another is not possible to estimate the probabilities of the states of nature. In this research, the stochastic knapsack problem with random capacity but known probability distribution for both discrete and continuous random capacity were studied.

The studied problem is the problem in which the allocation of the decision variables x_j are made to meet random capacity with a known distribution. The objective function in this problem may be a total cost, which can be divided into two parts. The first part is the cost of selecting the decision variables x_j and the second part is the penalty cost (i.e., shortage and storage cost). When the capacity has discrete probability distribution, the problem is called Stochastic Knapsack Problem with Discrete Random Capacity (SKPDRC). It can be stated as follows.

(1)

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} (gp_i u_i + hp_i v_i)$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u_i - v_i = b_i$$

$$0 \le x_j \le t_j \text{ and integer}$$

$$\sum_{i=1}^{m} p_i = 1$$

where x_j is the decision variable, for $j = 1, 2, ..., n$,

t_j is the upper bound of x_j, for $j = 1, 2, ..., n$,

u_i is the slack variable and $u_i \ge 0$, for $i = 1, 2, ..., m$,

v_i is the surplus variable and $v_i \geq 0$, for $i = 1, 2, ..., m$,

a_j is the weight coefficient of item j and $a_j \geq 0$, for $j = 1, 2, ..., n$,

c_j is the cost coefficient of item j and $c_j \geq 0$, for $j = 1, 2, ..., n$,

b_i is the capacity of alternative i and $b_i \geq 0$, for $i = 1, 2, ..., m$,

p_i is the probability of having capacity b_i and $p_i \geq 0$, for $i = 1, 2, ..., m$,

g is the per unit cost of having u_i, $g \geq 0$, and

h is the per unit cost of having v_i, $h \geq 0$.

When the capacity has continuous probability distribution, the problem is called Stochastic Knapsack Problem with Continuous Random Capacity (SKPCRC). It can be stated as follows.

(2)

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + g\int_L^U p(b)u(b)db + h\int_L^U p(b)v(b)db$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u(b) - v(b) = b$$

$$0 \leq x_j \leq t_j \text{ and integer}$$

$$\int_L^U p(b)db = 1$$

where x_j is the decision variable, for $j = 1, 2, ..., n$,

t_j is the upper bound of x_j, for $j = 1, 2, ..., n$,

$u(b)$ is the slack variable and $u(b) \geq 0$, for $L \leq b \leq U$,

$v(b)$ is the surplus variable and $v(b) \geq 0$, for $L \leq b \leq U$,

a_j is the weight coefficient of item j and $a_j \geq 0$, for $j = 1, 2, ..., n$,

c_j is the cost coefficient of item j and $c_j \geq 0$, for $j = 1, 2, ..., n$,

b is the capacity,

$p(b)$ is the probability of having capacity b, $p(b) \geq 0$, for $L \leq b \leq U$,

g is the per unit cost of having $u(b)$, $g \geq 0$, and

h is the per unit cost of having $v(b)$, $h \geq 0$.

OBJECTIVES

The objectives of this research are as follows.

1. To propose theory and methods for solving the relaxed problem of the stochastic knapsack problem with discrete and continuous random capacity.

2. To propose the heuristic for solving the stochastic knapsack problem with discrete and continuous random capacity.

3. For the stochastic knapsack problems with discrete random capacity, efficiency (i.e., computing time) of the proposed algorithms were compared with the general purpose method using CPLEX. Effectiveness (i.e., quality of integer solutions) of the proposed algorithms and proposed heuristic were compared with the general purpose method using CPLEX. For the stochastic knapsack problems with continuous random capacity, efficiency (i.e., computing time) and effectiveness (i.e., quality of solutions/integer solutions) of the proposed algorithms and the proposed heuristic were compared with the general purpose method using the Monte Carlo simulation.

LITERATURE REVIEW

1. Knapsack Problem

Knapsack problem is a special case of integer programming problem. To simplify knapsack problem, it is an integer linear programming problem that has only one constraint. It is the problem deciding which items with known benefit and weight should be loaded into a knapsack with fixed capacity. The objective is to maximize the total benefit without exceeding capacity constraint.

1.1 The 0-1 Knapsack Problem

Consider 0-1 knapsack problem, a subset of *n* given items has to be packed in a knapsack of capacity *C*. Each item has a profit p_j and a weight w_j and a problem is to select a subset of the items x_j whose total weight does not exceed capacity *C* and whose total profit is a maximum. The 0-1 knapsack problem can be stated as follows.

$$\text{Maximize } Y = \sum_{j=1}^{n} p_j x_j$$

$$\text{Subject to } \sum_{j=1}^{n} w_j x_j \le C$$

$$x_j \in \{0,1\},\ j = 1,2,...,n$$

The principal approaches for solving a 0-1 knapsack problem are dynamic programming approaches and branch and bound approaches. Examples of dynamic programming approaches for solving 0-1 knapsack problem can be found in Bellman (1957); Toth (1980); Pisinger (1997). Dynamic programming approach is not an efficient way to solve large size knapsack problem. Examples of branch and bound algorithms for solving 0-1 knapsack problem are presented in Kolesar (1967); Greenberg and Hegerich (1970); Horowitz and Sahni (1974); Fayard and Plateau (1975, 1982); Nauss (1976); Martello and Toth (1977a, 1988, 1990, 1997); Balas and Zemel (1980); Pandit and Ravi (1993); Pisinger (1995). Hybrid algorithms,

combining the dynamic programming and the branch and bound approaches have been presented in Martello and Toth (1984); Elkihel and Plateau (1985); Martello *et al*. (1999, 2000).

Greenberg and Hegerich (1970) presented two algorithms that are BS, which is branch and search algorithm, and BB, which is branch and bound algorithm, and compared them with BK, which is branch and bound algorithm presented in Kolesar (1967). The result indicated that BS is faster than BB and BK, and BB is faster than BK.

Ingargiola and Korsh (1973) developed a reduction algorithm for 0-1 knapsack problem, which has proved to be very effective in reducing computation times. The advantage of using reduction algorithm is reducing the size of before any search is undertaken. Time required for the reduction algorithm is linearly proportional to the number of variables.

There are effective algorithms based on a fixed core problem approach that are BZ (Balas and Zemel, 1980), FP (Fayard and Plateau, 1982), MT2 (Martello and Toth, 1988). Martello and Toth (1990) compared these core algorithms. The result is shown that MT2 algorithm give the best performance. Pisinger (1995) proposed expanding-core algorithm, called EXPKNAP, and compared with MT2 by using four types of randomly generated data instances that are uncorrelated, weakly correlated, strongly correlated, and subset-sum. As the result, EXPKNAP is better in case of uncorrelated, weakly correlated and subset-sum instances. However, MT2 is better for strongly correlated instances.

Martello and Toth (1997) proposed MTH algorithm, which is an effective branch and bound algorithm with tighter bound. In this algorithm, new upper bounds obtained from Lagrangian relaxation of cardinality constraints. They compared MTH algorithm with MT2 (Martello and Toth, 1988), MTR (Martello and Toth, 1990) + DT (Toth, 1980), and PR (Pandit and Ravi, 1993). As a result, PR and MTH are only two algorithms that can effectively solve strongly correlated and inverse strongly

correlated instances. The most efficient algorithm is PR for small number of decision variable and MTH for large number of decision variable. For almost strongly correlated and uncorrelated instances with similar weights, MTH is clearly the only effective algorithm. It is an effective algorithm for uncorrelated and weakly correlated instances, although MT2 is slightly better for uncorrelated and weakly correlated instances.

Pisinger (1997) proposed a dynamic programming algorithm, called MINKNAP. In this algorithm, a minimal core is determined by using expanding-core approach (EXPKNAP) that presented in Pisinger (1995). MINKNAP is compared with MT2 algorithm. As a result, MINKNAP can solve all instances but MT2 is not able to solve strongly correlated data instances of large problem size.

Martello and Toth (1999) proposed COMBO algorithm, which combines dynamic programming with tighter bounds. In this algorithm, new upper bounds obtained from the surrogate relaxation of cardinality constraints.

Martello *et al*. (2000) gave an overview of the recent techniques for solving hard knapsack problems that are basic branch and bound algorithms, core algorithms, dynamic programming algorithms, tighter bounds, and combined approach. They compared MT2, MINKNAP, MTH by using seven types of randomly generated data instances that are uncorrelated, weakly correlated, strongly correlated, inverse strongly correlated, almost strongly correlated, subset-sum, and uncorrelated with similar weights. As a result, MT2 algorithm cannot solve hard instances, but it perform well for uncorrelated, weakly correlated and subset-sum instances. MINKNAP algorithm has an overall stable behavior but MTH algorithm can solve faster than MINKNAP algorithm due to better upper bound. COMBO algorithm is superior to the others.

Ram and Sarin (1988) described branch and bound algorithm for the 0-1 equality knapsack problem. Marsman and Volgenant (1998) applied core approach to

the 0-1 equality knapsack problem, and compared with Ram and Sarin algorithm. They have shown that core approach is better in solution time.

1.2 Bounded knapsack problem (BKP)

Consider Bounded Knapsack Problem (BKP), *n* types of objects have to be put into a knapsack of capacity *C*. Each object of type *j* has a profit p_j and a weight w_j and we have a bounded number of copies of each object type t_j. We want to determine the number of x_j that maximize total profit without exceeding capacity *C* and its bound t_j. The BKP can be stated as follows.

$$\text{Maximize } Y = \sum_{j=1}^{n} p_j x_j$$

$$\text{Subject to } \sum_{j=1}^{n} w_j x_j \le C$$

$$0 \le x_j \le t_j \text{ and integer, } j = 1, 2, ..., n$$

BKP is an NP-hard problem. It can be solved by branch and bound approach (Ingargiola and Korsh, 1977; Martello and Toth, 1977b) and dynamic programming approach (Nemhauser and Ullmann, 1969; Pferschy, 1999). Martello and Toth (1990) presented MTB2, which transform BKP to the original 0-1 knapsack problem. The obtained KP with $\sum_{j=1}^{n} t_j$ variables is solved by using BALKNAP algorithm. Pisinger (2000) developed BOUKNAP algorithm that solves an expanding core problem through dynamic programming such that the number of enumerated item types is minimal. The BOUKNAP algorithm is compared with MTB2 (Matello and Toth, 1990). The result is shown that BOUKNAP is better than MTB2 for all kinds of random generated problem types.

When $t_j = \infty$, for $j = 1, 2, ..., n$, this problem is called unbounded knapsack problem.

1.3 Unbounded knapsack problem (UKP)

Consider Unbounded Knapsack Problem (UKP), *n* types of objects have to be put into a knapsack of capacity *C*. Each object of type *j* has a profit p_j and a weight w_j and we have an unbounded number of copies of each object type. We want to determine the number of x_j that maximize total profit without exceeding capacity *C*. The UKP can be expressed as follows.

$$\text{Maximize } Y = \sum_{j=1}^{n} p_j x_j$$

$$\text{Subject to } \sum_{j=1}^{n} w_j x_j \le C$$

$$x_j \ge 0 \text{ and integer, } j = 1, 2, ..., n$$

UKP is a classic NP-hard problem with a wide range of applications (Hu, 1969; Garfinkel and Nemhauser, 1972; Nemhauser and Wolsey, 1988; Martello and Toth, 1990). UKP may be solved as an ordinary BKP by imposing the bound $t_j = \lfloor C / w_j \rfloor$ on each item type *j*. The two classical approaches for solving this problem are branch and bound approach (Martello and Toth, 1990) and dynamic programming approach (Hu, 1969; Garfinkel and Nemhauser, 1972; Nemhauser and Wolsey, 1988; Andonov *et al*., 2000). Both of these two algorithms have a running time that is bounded by an exponential function of the length of input data, thus it is very difficult to obtain the exact solutions to many large-scale knapsack instances. Hence for those large-scale instances, it has to use a heuristic algorithm to obtain the near-optimal solutions to them. Genetic algorithm (GA) is an effective heuristic algorithm for solving knapsack problem. Guang *et al*. (2003) proposed GA for solving UKP. This algorithm can obtain more exact solutions for various problems and requires less computational time compared with EDUK (Andonov *et al*., 2000) and QGA (Han *et al*., 2001) solvers.

1.4 Mutidimensional Knapsack Problem (MKP)

Mutidimensional Knapsack Problem (MKP) has several names such as multiconstraint knapsack problem, *m*-dimensional knapsack problem and multiknapsack problem. The 0-1 mutidimensional knapsack problem is a generalization of 0-1 knapsack problem $(m = 1)$. It can be expressed as follows.

Maximize $Y = \sum_{j=1}^{n} p_j x_j$

Subject to $\sum_{j=1}^{n} w_{ij} x_j \le C_i$, $i = 1, 2, ..., m$

$x_j = \{0, 1\}$, $j = 1, 2, ..., n$

where n is the number of items and m is the number of knapsack constraints with capacity C_i.

Exact solving of 0-1 MKP is usually considered in a dynamic programming approach, branch and bound framework, and some schemes were designed to provide good lower and upper bounds. Gilmore and Gomory (1966) developed a dynamic programming for solving this problem. Marsten and Morin (1978) combined dynamic programming and branch and bound approach for solving this problem. Shih (1979) proposed a method to find a good upper bound. The thirty test problems with five knapsack constraints and number of variables ranging from 30 to 90 are randomly generated. This method can solve the tested problems faster than the original Balas and improved Balas additive algorithms (Balas *et al*., 1965). To find a good bound, Lagrangian and surrogate relaxation based on branch and bound methods have been developed. Early investigation with Lagrangian multipliers in solving this problem began with the paper of Lories and Savage (1955). They proposed Lagrangian heuristic for 0-1 integer programming, in which all of the constraints were relaxed in the objective function. Their approach was later formalized by Everett (1963) and extended by Kaplan (1966). The surrogate strategy was introduced by Glover (1965). The other papers related to surrogate relaxation are Glover (1968, 1975); Dyer (1980); Karwan and Radin (1984); Freville and Plateau (1993). Gavish and Pirkul (1985) developed branch and bound algorithm embedding

new algorithms which are surrogate bounds and rules for reducing problem size. As the result, their method was significantly faster than the method developed by Shih (1979). Pisinger (1999a) developed MULKNAP algorithm for solving 0-1 MKP, which is improved from MTM algorithm (Martello and Toth, 1981) in several aspects. The lower bounds are obtained by splitting the surrogate solution into *m* knapsacks by solving a series of subset-sum problems. Furthermore, subset-sum problems are also used for tightening the capacity constraints of each knapsack in order to obtain better upper bounds. The developed algorithm is compared to MTM algorithm by using four types of randomly generated data instances that are uncorrelated, weakly correlated, strongly correlated, and subset-sum instances. As a result, MULKNAP can solve all instances faster than MTM. Moreover, MTM cannot solve hard problems (i.e., strongly correlated instances of large size).

Due to intrinsic difficulty of 0-1 MKP for larger instances, several heuristic have been used to solve it such as simulated annealing (Drexl, 1988), tabu search (Glover and Kochenberger, 1996; Freville and Hanafi, 1998) and genetic algorithms (Beasly and Chu, 1998).

2. Stochastic Programming

Stochastic programming is a framework for modeling optimization problems that involve uncertainty. It was described by Kall and Wallace (1994); Birge and Louveaux (1997). Two basic models of stochastic program are chance constraint model and two-stage model. For two-stage program, the decision maker makes a decision in the first stage then the random effects occur. After that the decision maker can make a second-stage decision that compensates for any bad effects from the first-stage decision. For chance constraint model, it does not require that our decisions are feasible for every outcome of the random parameters, but require feasibility with at least some specified probability. The stochastic program with recourse was first considered by Dantzig (1955) under the name two-stage linear programs under uncertainty.

Approaches for solving stochastic program can be classified into three groups that are exact solution procedures (i.e., simplex method, interior point algorithm, and decomposition methods), bounding and approximation schemes (i.e., lower bounds (e.g., Jensen's inequality), upper bound (e.g., Edmundson-Madansky inequality (Edmundson, 1956; Madansky, 1959), and apply bounds conditionally to partitions of Ξ), Monte Carlo sampling based-methods (e.g., stochastic quasi gradient methods (Ermoliev, 1983), importance sampling method (Dantzig and Glynn, 1990) and stochastic decomposition methods (Higle and Sen, 1991).

The algorithms developed for solving stochastic program with recourse was presented by Elmaghraby (1959, 1960); Ziemba (1970); Garstka and Rutenberg (1970); Audsley *et al*. (2000).

Elmaghraby (1959) presented that production problem with stochastic demand, when distribution function demand is discrete, can be transformed to be the linear programming problem. The objective function is minimizing production cost plus storage cost plus shortage cost. He proved that the objective function is piecewise convex function. To transform the stochastic problem to be linear programming problem, the objective function of minimizing cost is changed to be minimizing partial differentiate of original objective function. Moreover, he gave a numerical example of transforming problem. He also studied this problem when the demand is continuous (Elmaghraby, 1960).

Ziemba (1970) modified two algorithms that are Frank and Wolfe algorithm (Frank and Wolfe, 1956) and convex simplex method (Zangwill, 1967) for solving convex stochastic program with simple recourse.

Garstka and Rutenberg (1970) presented a procedure for solving discrete stochastic programs with recourse. It views the *m* stochastic elements of the right-hand side vector as an *m*-dimensional space in which each combination of the discrete values is a lattice point. For a given second stage basis, certain of lattice points are feasible. A procedure is presented to delete infeasible points from this *m*-dimensional

space until only feasible points remain. Thus, the probability associated with feasible lattice for this basis can be enumerated, and used to weight the vector of dual variables defined by the current optimal basis. Finally, they presented a systematic procedure for changing optimal basis so that a feasible and optimal basis is found for every lattice point.

Audsley *et al*. (2000) presented a two-stage stochastic programming with recourse model for the problem of determining optimal planting plans for a vegetable crop. The first stage of the model relates to finding a planting plan. The second stage is concerned with deriving a harvest schedule for each scenario. They solved this problem by using decomposition technique, and compared their solutions with the solutions from deterministic equivalent model, which is solved by using interior point method. Their solutions were greater expected profit than the deterministic model and also more robust.

Ermoliev *et al*. (1998) presented a stochastic branch and bound method for solving stochastic discrete programming problem. In this method, stochastic lower and upper bounds are calculated and combined with branch and bound method. In order to find stochastic lower and upper bounds, they applied two general ideas which are interchange of minimization and mathematical expectation operators, and dual estimates. As a result, when using stochastic bounds, the optimal solution will be reached faster than using deterministic bounds. There are other researches relating stochastic discrete programming presented by Mukai and Yan (1992); Andradottir (1996). These researches used random search techniques which aim at finding a nearly-optimal solution.

3. Stochastic Knapsack Problem (SKP)

In realistic situations, the benefit, weight and/or capacity in the knapsack problem may not be known with certainty, which is called Stochastic Knapsack Problem (SKP) can be classified as static and dynamic. In Static Stochastic Knapsack Problem (SSKP), the set of items is given, but there is uncertainty in the coefficients

(e.g., benefit, weight, and/or capacity coefficients). SSKP were investigated by Parks and Steinberg (1979); Sniedovich (1980); Henig (1990). In Dynamic Stochastic Knapsack Problem (DSKP), the item arriving is random, and the benefit and/or weight coefficients are unknown before item arrival. DSKP has various application such as resource allocation problem (Kleywegt *et al*., 1996; Kleywegt and Papastavrou, 1998, 2001), project selection (Chiu *et al*., 1999), and yield management (Slyke and Young, 2000).

Parks and Steinberg (1979); Sniedovich (1980) are solving SSKP when benefit coefficients are uncertain by transforming this problem to a preference ordering problem and applying dynamic programming approach for solving it. Sniedovich (1980) proposed the strong criteria and indicated that the weak criteria proposed by Parks and Steinberg (1979) may not give an optimal solution.

Henig (1990) presented an alternative way to solve SSKP with uncertainty in benefit coefficients. He combined dynamic programming approach with branch and bound method in order to maximize the probability of attaining the objective value.

Kleywegt *et al*. (1996) studied resource allocation problem when item arrival is random and it has Poisson distributed. Both weights and benefits are random and they become known as soon as the items arrive. They assumed that there are no holding costs. Each item must be accepted or rejected as soon as it arrives. The objective is to determine the optimal policy for allocating the capacity within a finite time horizon so as to maximize the expected sum of benefits of accepted items.

Kleywegt and Papastavrou (1998) also studied resource allocation problem when item arrival is random and it has Poisson distributed. In this research, it is assumed that items have equal weight, and benefits of items are random and become known as soon as the items arrive. Each item must be accepted or rejected as soon as it arrives. They studied DSKP with holding costs for both finite and infinite time horizon cases.

Kleywegt and Papastavrou (2001) also studied resource allocation problem when item arrival is random and it has Poisson distributed. And both benefit and weight are random and they become known as soon as the items arrive. Each item must be accepted or rejected as soon as it arrives. They studied DSKP with holding costs for both finite and infinite time horizon cases.

Chiu *et al*. (1999) studied project selection problem when project arrival is random with known probability distribution. And both benefits and weights are random and become known as soon as the projects arrive. Each project must be accepted or rejected as soon as it arrives. The problem is to determine the optimal policy for allocating a budget to projects over time so as to maximize the expected benefit within a finite time horizon. Furthermore, they considered the case where time horizon is an exponential random variable.

Slyke and Young (2000) studied yield management problem when the customer arrival is random and it is Poisson distributed. Both weights and benefits of customers are random and they become known as soon as customers arrive. Each customer must be accepted or rejected as soon as customer arrives. The problem is to determine the optimal policy for allocating a capacity to serve the customer over time so as to maximize the expected sum of benefit of the accepted customers within a finite time horizon.

Since, there are few researchers concerning with solving static stochastic knapsack problem when the benefit is uncertain. However, studying uncertainty in capacity is also important. Thus, this research will be focused on static stochastic knapsack problem where the capacity is uncertain for both discrete and continuous random capacity.

MATERIALS AND METHODS

Materials

1. A personal computer, Intel Pentium M Processor 1.6 GHz., 512 MB RAM, 60 GB hard drive
2. MATLAB 7.0.1 software
3. CPLEX 8.1 software
4. Visual C.NET software

Methods

In this section, the methods for solving SKPDRCR and SKPCRCR were presented. Finally, the heuristics for solving SKPDRC and SKPCRC were proposed.

1. Relaxed Problem of Stochastic Knapsack Problem with Discrete Random Capacity (SKPDRCR)

According to (1), if we relaxed integer constraint, then the relaxed problem is SKPDRCR that can be stated as follows.

(3)

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} (gp_i u_i + hp_i v_i)$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u_i - v_i = b_i$$

$$0 \le x_j \le t_j$$

$$\sum_{i=1}^{m} p_i = 1$$

where x_j is the decision variable, for $j = 1, 2, ..., n$,

t_j is the upper bound of x_j, for $j = 1, 2, ..., n$,

u_i is the slack variable and $u_i \ge 0$, for $i = 1, 2, ..., m$,

v_i is the surplus variable and $v_i \geq 0$, for $i = 1, 2, ..., m$,

a_j is the weight coefficient of item j and $a_j \geq 0$, for $j = 1, 2, ..., n$,

c_j is the cost coefficient of item j and $c_j \geq 0$, for $j = 1, 2, ..., n$,

b_i is the capacity of alternative i and $b_i \geq 0$, for $i = 1, 2, ..., m$,

p_i is the probability of having capacity b_i and $p_i \geq 0$, for $i = 1, 2, ..., m$,

g is the per unit cost of having u_i, $g \geq 0$, and

h is the per unit cost of having v_i, $h \geq 0$.

We proposed two algorithms for solving SKPDRCR. In order to easily explain these algorithms, we have two assumptions that are $b_i \leq b_{i+1}$, for $i = 1, 2, ..., m-1$, and $c_j / a_j \leq c_{j+1} / a_{j+1}$, for $j = 1, 2, ..., n-1$.

1.1 Algorithm 1 of SKPDRCR

There exist four optimality conditions. For each case, we proved by writing basic variables in terms of nonbasic variables. Then substitute basic variables into objective function. The optimal solution can be found when the reduced cost of nonbasic variables are greater than or equal to zero.

Case I: the optimal solution is as follows.

$x_j = 0, \forall j$

$v_i = 0, \forall i$

$u_i = b_i, \forall i$

where $c_1 / a_1 \geq g$.

Proof

$$u_i = b_i - \sum_{j=1}^{n} a_j x_j + v_i, \; i = 1, 2, ..., m$$

Next part is to substitute these basic variables into the objective function as follows.

$$f = \sum_{j=1}^{n} c_j x_j + g\sum_{i=1}^{m} p_i(b_i - \sum_{j=1}^{n} a_j x_j + v_i) + h\sum_{i=1}^{m} p_i v_i$$
$$= g\sum_{i=1}^{m} p_i b_i + \sum_{i=1}^{m}(g+h)p_i v_i + \sum_{j=1}^{n}(c_j - ga_j\sum_{i=1}^{m} p_i)x_j$$
$$= g\sum_{i=1}^{m} p_i b_i + \sum_{i=1}^{m}(g+h)p_i v_i + \sum_{j=1}^{n}(c_j - ga_j)x_j$$

Since, the minimize f can be found when all reduced costs of nonbasic variables are greater than or equal to zero. In this case, nonbasic variables are x_j, for all j and v_i, for all i. In order to obtain the minimum value of *f*, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $(g+h)p_i \geq 0$, for all i and $(c_j - ga_j) \geq 0$, for all j.

Since $g, h \geq 0$ and $p_i \geq 0$, for all *i*, $(g+h)p_i \geq 0$, for all *i*. To minimize *f*, $(c_j - ga_j) \geq 0$, for all *j*, must be greater than or equal to zero that is equivalent to $c_j / a_j \geq g$, for all j. According to the assumption $c_j / a_j \leq c_{j+1} / a_{j+1}$, for $j = 1, 2, ..., n-1$, the condition for this case is $c_1 / a_1 \geq g$. In this case, the minimum value of f is $g\sum_{i=1}^{m} p_i b_i$.

Case II: the optimal solution is as follows.

$x_j = t_j$, $\forall j$

$v_i = \sum_{j=1}^{n} a_j t_j - b_i$ and $u_i = 0, \quad i \in I_2$

$u_i = b_i - \sum_{j=1}^{n} a_j t_j$ and $v_i = 0, \quad i \in I_1$

where $I_1 = \{i = 1 : m : b_i \geq \sum_{j=1}^{n} a_j t_j\}$, $I_2 = \{i = 1 : m : b_i < \sum_{j=1}^{n} a_j t_j\}$, and

$c_n / a_n \leq g\sum_{i \in I_1} p_i - h\sum_{i \in I_2} p_i$.

Proof

$x_j = t_j - r_j$, $\forall j$

$$r_j \geq 0,\ \forall j$$

$$u_i = b_i - \sum_{j=1}^{n} a_j t_j + \sum_{j=1}^{n} a_j r_j + v_i,\ i \in I_1$$

$$v_i = -b_i + \sum_{j=1}^{n} a_j t_j - \sum_{j=1}^{n} a_j r_j + u_i,\ \ i \in I_2$$

Next part is to substitute these basic variables into the objective function as follows.

$$f = \sum_{j=1}^{n} c_j (t_j - r_j) + g \sum_{i \in I_1} p_i (b_i - \sum_{j=1}^{n} a_j t_j + \sum_{j=1}^{n} a_j r_j + v_i) + h \sum_{i \in I_1} p_i v_i$$
$$+ h \sum_{i \in I_2} p_i (-b_i + \sum_{j=1}^{n} a_j t_j - \sum_{j=1}^{n} a_j r_j + u_i) + g \sum_{i \in I_2} p_i u_i$$

$$= \sum_{j=1}^{n} c_j t_j + g \sum_{i \in I_1} p_i (b_i - \sum_{j=1}^{n} a_j t_j) + h \sum_{i \in I_2} p_i (-b_i + \sum_{j=1}^{n} a_j t_j)$$
$$+ \sum_{i \in I_1} (g+h) p_i v_i + \sum_{i \in I_2} (g+h) p_i u_i + \sum_{j=1}^{n} (-c_j + g a_j \sum_{i \in I_1} p_i - h a_j \sum_{i \in I_2} p_i) r_j$$

In this case, nonbasic variables are v_i, for $i \in I_1$, u_i, for $i \in I_2$ and r_j, for all j. In order to obtain the minimum value of *f*, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $(g+h) p_i \geq 0$, for $i \in I_1$, $(g+h) p_i \geq 0$, for $i \in I_2$ and $(-c_j + g a_j \sum_{i \in I_1} p_i - h a_j \sum_{i \in I_2} p_i) \geq 0$, for all j

Since $g, h \geq 0$ and $p_i \geq 0$, for all i, $(g+h) p_i \geq 0$, for $i \in I_1$, $(g+h) p_i \geq 0$, for $i \in I_2$ have been approved. To minimize *f*, $(-c_j + g a_j \sum_{i \in I_1} p_i - h a_j \sum_{i \in I_2} p_i)$, for all j must be greater than or equal to zero that is equivalent to $c_j / a_j \leq g \sum_{i \in I_1} p_i - h \sum_{i \in I_2} p_i$, for all j. According to the assumption $c_j / a_j \leq c_{j+1} / a_{j+1}$, for $j = 1, 2, ..., n-1$, the condition for this case is

$c_n / a_n \le g\sum_{i\in I_1} p_i - h\sum_{i\in I_2} p_i$. In this case, the minimum value of f is $\sum_{j=1}^{n} c_j t_j$

$+g\sum_{i\in I_1} p_i(b_i - \sum_{j=1}^{n} a_j t_j) + h\sum_{i\in I_2} p_i(-b_i + \sum_{j=1}^{n} a_j t_j)$.

Case III: the optimal solution is as follows.

$$x_j = t_j, \quad j = 1,...,k-1$$

$$x_k = \frac{b_q - \sum_{j=1}^{k-1} a_j t_j}{a_k}$$

$$x_j = 0, \quad j = k+1,...,n$$

$$v_i = b_q - b_i \text{ and } u_i = 0, \qquad i = 1,2,...,q-1$$

$$u_q = 0 \text{ and } v_q = 0$$

$$u_i = b_i - b_q \text{ and } v_i = 0, \qquad i = q+1, q+2,...,m$$

where $q \in \{1,...,m\}$ and $k \in \{1,...,n\}$ such that

$(-c_k / a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i) \ge 0$, $(c_k / a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i) \ge 0$, $\sum_{j=1}^{k-1} a_j t_j \le b_q$ and

$\sum_{j=1}^{k} a_j t_j \ge b_q$. If $\sum_{j=1}^{n} a_j t_j < b_q$, then $k = n$.

Proof

Suppose *k* and *q* have been specified.

$$f(q) = \sum_{j=1}^{k-1} c_j x_j + c_k x_k + \sum_{j=k+1}^{n} c_j x_j + g\sum_{i=1}^{q-1} p_i u_i + h\sum_{i=1}^{q-1} p_i v_i + g\sum_{i=q+1}^{m} p_i u_i$$
$$+h\sum_{i=q+1}^{m} p_i v_i + g p_q u_q + h p_q v_q \qquad (3.1)$$

Let,

$x_j = t_j, j = 1,...,k-1,$

$$x_k = (b_q - u_q + v_q - \sum_{j=1}^{k-1} a_j x_j - \sum_{j=k+1}^{n} a_j x_j) / a_k$$

$$x_k = (b_q - u_q + v_q - \sum_{j=1}^{k-1} a_j t_j - \sum_{j=k+1}^{n} a_j x_j)/a_k ,$$

$$\begin{aligned} v_i &= \sum_{j=1}^{k-1} a_j x_j + a_k x_k + \sum_{j=k+1}^{n} a_j x_j + u_i - b_i \\ &= \sum_{j=1}^{k-1} a_j x_j + a_k((b_q - u_q + v_q - \sum_{j=1}^{k-1} a_j x_j - \sum_{j=k+1}^{n} a_j x_j)/a_k) + \sum_{j=k+1}^{n} a_j x_j + u_i - b_i \\ &= b_q - b_i + u_i - u_q + v_q ,\ i = 1,2,...,q-1 , \text{ and} \end{aligned}$$

$$\begin{aligned} u_i &= b_i + v_i - \sum_{j=1}^{k-1} a_j x_j - a_k x_k - \sum_{j=k+1}^{n} a_j x_j \\ &= b_i + v_i - \sum_{j=1}^{k-1} a_j x_j - a_k((b_q - u_q + v_q - \sum_{j=1}^{k-1} a_j x_j - \sum_{j=k+1}^{n} a_j x_j)/a_k) - \sum_{j=k+1}^{n} a_j x_j \\ &= b_i - b_q + v_i + u_q - v_q ,\ i = q+1, q+2,...,m . \end{aligned}$$

The next part is to substitute these basic variables into equation (3.1). Thus equation (3.1) becomes as follows.

$$\begin{aligned} f(q) &= \sum_{j=1}^{k-1} c_j t_j + c_k((b_q - u_q + v_q - \sum_{j=1}^{k-1} a_j t_j - \sum_{j=k+1}^{n} a_j x_j)/a_k) + \sum_{j=k+1}^{n} c_j x_j \\ &\quad + g\sum_{i=1}^{q-1} p_i u_i + h\sum_{i=1}^{q-1} p_i(b_q - b_i + u_i - u_q + v_q) \\ &\quad + g\sum_{i=q+1}^{m} p_i(b_i - b_q + v_i + u_q - v_q) + h\sum_{i=q+1}^{m} p_i v_i + g p_q u_q + h p_q v_q \\ &= \sum_{j=1}^{k-1} c_j t_j + c_k((b_q - \sum_{j=1}^{k-1} a_j t_j)/a_k) + h\sum_{i=1}^{q-1} p_i(b_q - b_i) + g\sum_{i=q+1}^{m} p_i(b_i - b_q) \\ &\quad + \sum_{j=k+1}^{n} (c_j - c_k a_j/a_k) x_j + \sum_{i=1}^{q-1} (g+h) p_i u_i + \sum_{i=q+1}^{m} (g+h) p_i v_i \\ &\quad + (-c_k/a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i) u_q + (c_k/a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i) v_q \end{aligned}$$

In this case, nonbasic variables are x_j, for $j = k+1, k+2, ..., n$, u_i, for $i = 1, 2, ..., q-1$, v_i, for $i = q+1, q+2, ...m$, u_q and v_q. In order to obtain the minimum value of *f*, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $c_j - c_k a_j/a_k \geq 0$, for $j = k+1, k+2, ..., n$, $(g+h)p_i \geq 0$, for

$i=1,2,...,q-1$, $(g+h)\,p_i \geq 0$, for $i=q+1,q+2,...,m$, $(-c_k/a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i) \geq 0$ and $(c_k/a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i) \geq 0$.

From assumption, $c_j/a_j \leq c_{j+1}/a_{j+1}$, for $j = 1, 2,..., n-1$, then $c_j - c_k a_j/a_k \geq 0$, for $j = k+1, k+2,..., n$. Since $g, h \geq 0$ and $p_i \geq 0$, for all i, $(g+h)p_i \geq 0$, for $i=1,2,...,q-1$, and $(g+h)\,p_i \geq 0$, for $i=q+1,q+2,...,m$ have been approved.

To minimize f, $(-c_k/a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i)$ and $(c_k/a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$ must be greater than or equal to zero. Since $0 \leq x_k \leq t_k$, it is equivalent to $0 \leq \dfrac{b_q - \sum_{j=1}^{k-1} a_j t_j}{a_k} \leq t_k$. It is also equivalent to $\sum_{j=1}^{k-1} a_j t_j \leq b_q$ and $\sum_{j=1}^{k} a_j t_j \geq b_q$. If $\sum_{j=1}^{n} a_j t_j < b_q$, then $k = n$.

Therefore, we select k and q such that $(-c_k/a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i) \geq 0$ and $(c_k/a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i) \geq 0$ and $\sum_{j=1}^{k-1} a_j t_j \leq b_q$ and $\sum_{j=1}^{k} a_j t_j \geq b_q$. If $\sum_{j=1}^{n} a_j t_j < b_q$, then $k = n$. In this case, the minimum value of f is $\sum_{j=1}^{k-1} c_j t_j + c_k((b_q - \sum_{j=1}^{k-1} a_j t_j)/a_k) + h\sum_{i=1}^{q-1} p_i(b_q - b_i) + g\sum_{i=q+1}^{m} p_i(b_i - b_q)$.

Case IV: the optimal solution is as follows.

$x_j = t_j$, $j = 1,...,k$

$x_j = 0$, $j = k+1,...,n$

$v_i = b^* - b_i$ and $u_i = 0$, $\quad i = 1, 2, ..., q$

$u_i = b_i - b^*$ and $v_i = 0$, $\quad i = q+1, q+2, ..., m$

where $q \in \{1, ..., m\}$ and $k \in \{1, ..., n\}$ such that

$$(c_{k+1}/a_{k+1} + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i) \geq 0,\ \sum_{j=1}^{k} a_j t_j > b_q,\ \sum_{j=1}^{k} a_j t_j < b_{q+1} \text{ and } b^* = \sum_{j=1}^{k} a_j t_j .$$

Proof

Suppose k and q have been specified.

$$f(q) = \sum_{j=1}^{k} c_j x_j + \sum_{j=k+1}^{n} c_j x_j + g\sum_{i=1}^{q} p_i u_i + h\sum_{i=1}^{q} p_i v_i + g\sum_{i=q+1}^{m} p_i u_i$$

$$+ h\sum_{i=q+1}^{m} p_i v_i \qquad (3.2)$$

Let,

$x_j = t_j,\ j = 1, ..., k$,

$$v_i = \sum_{j=1}^{k} a_j x_j + \sum_{j=k+1}^{n} a_j x_j + u_i - b_i$$

$$= \sum_{j=1}^{k} a_j t_j + \sum_{j=k+1}^{n} a_j x_j + u_i - b_i,\ i = 1, 2, ..., q, \text{ and}$$

$$u_i = b_i + v_i - \sum_{j=1}^{k} a_j x_j - \sum_{j=k+1}^{n} a_j x_j$$

$$= b_i + v_i - \sum_{j=1}^{k} a_j t_j - \sum_{j=k+1}^{n} a_j x_j,\ i = q+1, q+2, ..., m .$$

The next part is to substitute these basic variables into equation (3.2). Thus equation (3.2) becomes as follows.

$$f(q) = \sum_{j=1}^{k} c_j t_j + \sum_{j=k+1}^{n} c_j x_j + g\sum_{i=1}^{q} p_i u_i + h\sum_{i=1}^{q} p_i (\sum_{j=1}^{k} a_j t_j - b_i + u_i + \sum_{j=k+1}^{n} a_j x_j)$$

$$+ g\sum_{i=q+1}^{m} p_i (b_i - \sum_{j=1}^{k} a_j t_j + v_i - \sum_{j=k+1}^{n} a_j x_j) + h\sum_{i=q+1}^{m} p_i v_i$$

$$f(q)=\sum_{j=1}^{k}c_j t_j+h\sum_{i=1}^{q}p_i(\sum_{j=1}^{k}a_j t_j-b_i)+g\sum_{i=q+1}^{m}p_i(b_i-\sum_{j=1}^{k}a_j t_j)$$

$$+\sum_{i=1}^{q}(g+h)p_i u_i+\sum_{i=q+1}^{m}(g+h)p_i v_i+\sum_{j=k+1}^{n}(c_j+ha_j\sum_{i=1}^{q}p_i-ga_j\sum_{i=q+1}^{m}p_i)x_j$$

In this case, nonbasic variables are x_j for $j=k+1,k+2,...,n$, u_i for $i=1,2,...,q$, v_i, for $i=q+1,q+2,...m$. In order to obtain the minimum value of *f*, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $(g+h)\,p_i \geq 0$, for $i = 1, 2, ..., q$, $(g+h)\,p_i \geq 0$, for $i = q+1, q+2, ..., m$ and $(c_j+ha_j\sum_{i=1}^{q}p_i-ga_j\sum_{i=q+1}^{m}p_i)\geq 0$, for $j=k+1,k+2,...,n$.

Since $g,h\geq 0$ and $p_i\geq 0$, for all *i*, $(g+h)\,p_i \geq 0$, for $i = 1, 2, ..., q$ and $(g+h)\,p_i \geq 0$, for $i = q+1, q+2, ..., m$ have been approved. To minimize *f*, $(c_j+ha_j\sum_{i=1}^{q}p_i-ga_j\sum_{i=q+1}^{m}p_i)$, for $j=k+1,k+2,...,n$, must be greater than or equal to zero that is equivalent to $(c_j/a_j+h\sum_{i=1}^{q}p_i-g\sum_{i=q+1}^{m}p_i)\geq 0$, for $j = k+1, k+2,..., n$. According to the assumption $c_j/a_j \leq c_{j+1}/a_{j+1}$, for $j=1,2,...,n-1$, the condition for this case is $(c_{k+1}/a_{k+1}+h\sum_{i=1}^{q}p_i-g\sum_{i=q+1}^{m}p_i)\geq 0$. Since $b_q<b^*<b_{q+1}$ and $b^*=\sum_{j=1}^{k}a_j t_j$, it is equivalent to $\sum_{j=1}^{k}a_j t_j>b_q$ and $\sum_{j=1}^{k}a_j t_j<b_{q+1}$.

Therefore, we select *k* and *q* such that $(c_{k+1}/a_{k+1}+h\sum_{i=1}^{q}p_i-g\sum_{i=q+1}^{m}p_i)\geq 0$, $\sum_{j=1}^{k}a_j t_j>b_q$ and $\sum_{j=1}^{k}a_j t_j<b_{q+1}$. In this case, the minimum value of f is $\sum_{j=1}^{k}c_j t_j$ $+h\sum_{i=1}^{q}p_i(\sum_{j=1}^{k}a_j t_j-b_i)+g\sum_{i=q+1}^{m}p_i(b_i-\sum_{j=1}^{k}a_j t_j)$.

1.2 Algorithm 2 of SKPDRCR

This algorithm results in the optimal solution of the SKPDRCR by using the graphical method, which was recommended by Associate Professor Kamlesh Mathur from Case Western Reserve University. The relaxed problem (3) can be written as follows.

(4)

$$\underset{y,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c'_j y_j + \sum_{i=1}^{m} (gp_i u_i + hp_i v_i) \tag{4.1}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} y_j + u_i - v_i = b_i \tag{4.2}$$

$$0 \le y_j \le t'_j \tag{4.3}$$

$$\sum_{i=1}^{m} p_i = 1 \tag{4.4}$$

where $y_j = a_j x_j$,

$c'_j = \dfrac{c_j}{a_j}$, and

$t'_j = a_j t_j$.

Objective function (4.1) can be separated into two parts as follows.

$$\underset{\theta}{\text{Minimize}}\ f = H_1(\theta) + H_2(\theta)$$

where $H_1(\theta) = \sum_{j=1}^{n} c'_j y_j$, and $H_2(\theta) = \sum_{i=1}^{m} (gp_i u_i + hp_i v_i)$.

Graphs of $H_1(\theta)$ and $H_2(\theta)$ of SKPDRCR were shown in Figures 1 and 2, respectively.

Let $\sum_{j=1}^{jj} y_j = \theta_{jj}$, for $jj = 1, 2, ..., n$.

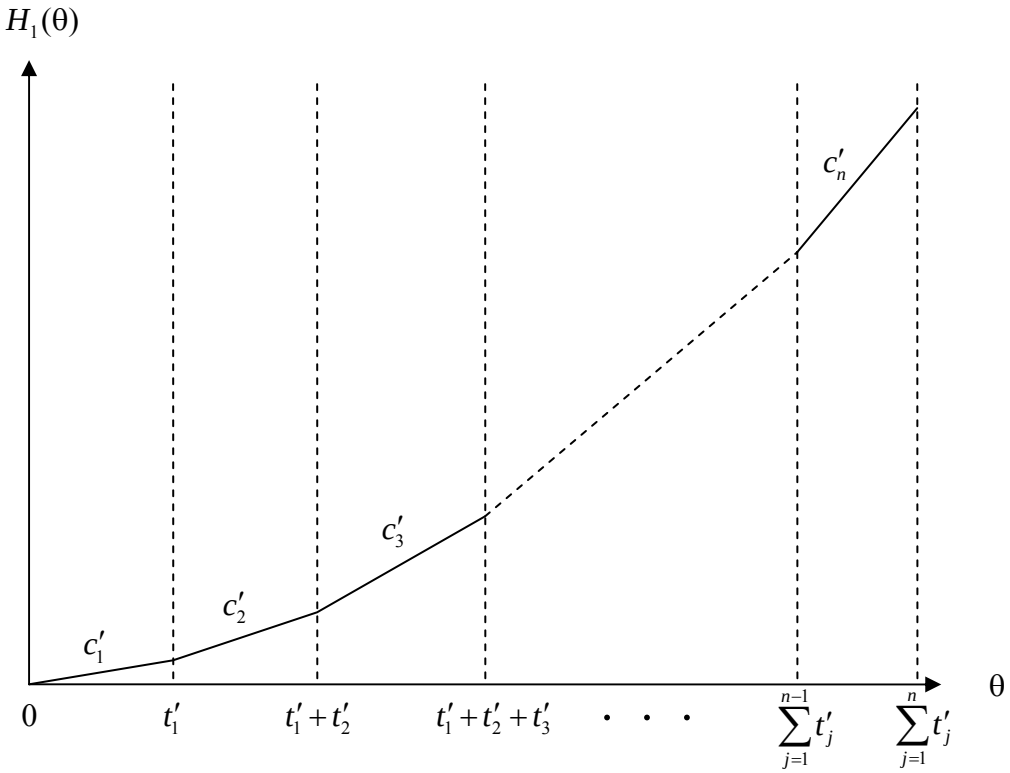

Figure 1 Graph of $H_1(\theta)$ of SKPDRCR

Let $SH1_{(t'_{j-1}\le\theta\le t'_j)}$ = slope of graph $H_1(\theta)$, $t'_{j-1} \le \theta \le t'_j$.

$= c'_j$

Since $c'_j \le c'_{j+1}$, for $j = 1,...n-1$, $H_1(\theta)$ is a piecewise-linear convex function.

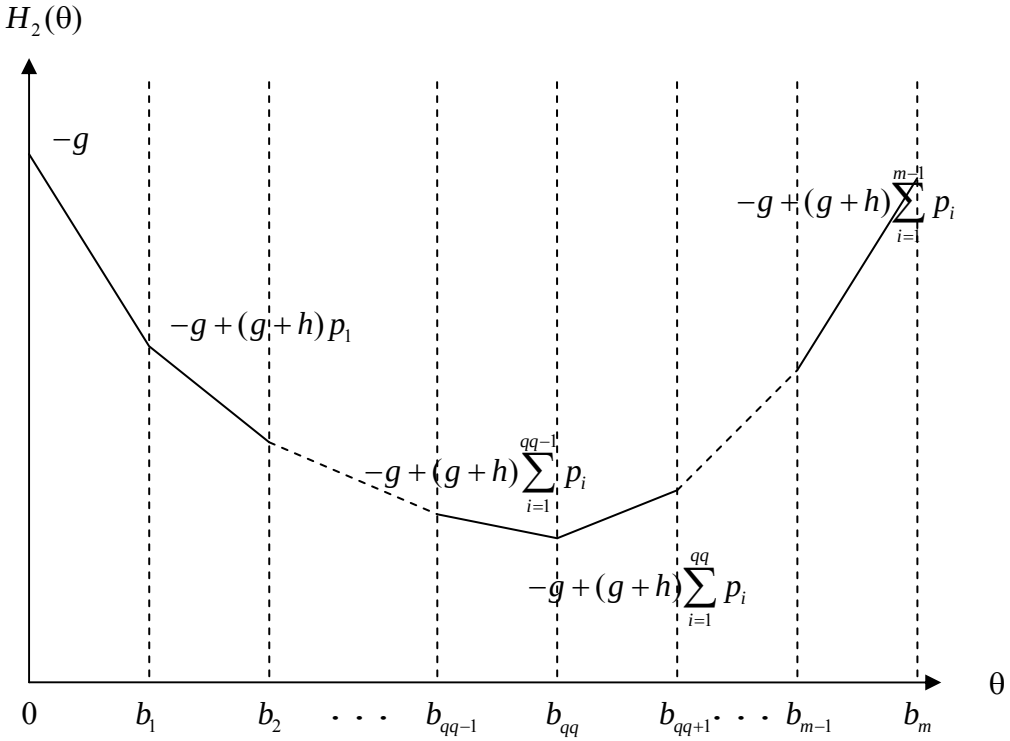

Figure 2 Graph of $H_2(\theta)$ of SKPDRCR

Let $SH2_{(b_{i-1}<\theta<b_i)} = \text{slope of graph } H_2(\theta),\ b_{i-1} < \theta < b_i$.

For $0 < \theta < b_1$,

$u_i = b_i - \theta \quad ,i = 1,...,m$, and

$v_i = 0 \quad ,i = 1,...,m$.

$$H_2(\theta) = \sum_{i=1}^{m} gp_i(b_i - \theta)$$

$$SH2_{(0\le\theta\le b_1)} = -g$$

For $b_1 < \theta < b_2$,

$v_1 = \theta - b_1$,

$u_i = b_i - \theta \quad ,i = 2,...,m$,

$v_i = 0 \quad ,i = 2,...,m$, and

$u_1 = 0$.

$$H_2(\theta) = \sum_{i=2}^{m} gp_i(b_i - \theta) + hp_1(\theta - b_1)$$

$$= \sum_{i=1}^{m} gp_i(b_i - \theta) + gp_1(\theta - b_1) + hp_1(\theta - b_1)$$

$$SH2_{(b_1 \le \theta \le b_2)} = -g + (g+h)p_1$$

For $b_{qq-1} < \theta < b_{qq}$,

$v_i = \theta - b_i$, $i = 1,...,qq-1$,

$u_i = b_i - \theta$, $i = qq,...,m$,

$v_i = 0$, $i = qq,...,m$, and

$u_i = 0$, $i = 1,...,qq-1$.

$$H_2(\theta) = \sum_{i=qq}^{m} gp_i(b_i - \theta) + \sum_{i=1}^{qq-1} hp_i(\theta - b_i)$$

$$= \sum_{i=1}^{m} gp_i(b_i - \theta) + \sum_{i=1}^{qq-1} gp_i(\theta - b_i) + \sum_{i=1}^{qq-1} hp_i(\theta - b_i)$$

$$SH2_{(b_{qq-1} < \theta < b_{qq})} = -g + (g+h)\sum_{i=1}^{qq-1} p_i$$

For $b_{qq} < \theta < b_{qq+1}$,

$v_i = \theta - b_i$, $i = 1,...,qq$,

$u_i = b_i - \theta$, $i = qq+1,...,m$,

$v_i = 0$, $i = qq+1,...,m$, and

$u_i = 0$, $i = 1,...,qq$.

$$H_2(\theta) = \sum_{i=qq+1}^{m} gp_i(b_i - \theta) + \sum_{i=1}^{qq} hp_i(\theta - b_i)$$

$$= g\sum_{i=1}^{m} p_i(b_i - \theta) + \sum_{i=1}^{qq} gp_i(\theta - b_i) + \sum_{i=1}^{qq} hp_i(\theta - b_i)$$

$$SH2_{(b_{qq} < \theta < b_{qq+1})} = -g + (g+h)\sum_{i=1}^{qq} p_i$$

Since $SH2_{(b_{i-1} < \theta < b_i)} \le SH2_{(b_i < \theta < b_{i+1})}$, for $i = 1,...m-1$, $H_2(\theta)$ is a piecewise convex function.

Since $H_1(\theta)$ is a piecewise-linear convex function and $H_2(\theta)$ is a piecewise convex function, $H_1(\theta)+H_2(\theta)$ is also a piecewise convex function.

Let SH_θ = slope of graph $H_1(\theta)+H_2(\theta)$.

There are $m+n+1$ extreme points. Therefore, there are $m+n$ slopes. The optimal solution can be found easily by calculating SH_θ from left to right. If SH_θ is greater than zero, the optimal solution is the starting point of that slope.

In this study, the algorithms 1 and 2 of SKPDRCR was written in MATLAB software as M-file program and compared with the general purpose method using CPLEX 8.1 (2002), which is the license of Case Western Reserve University. My deepest gratitude to Associate Professor Vira Chankong for allowing me to use this software. For the algorithm 1 of SKPDRCR, it can be generalized step by step as follows.

Step 1 Input n, m,

c_j, a_j, t_j, $j = 1,2,...,n$,

b_i, p_i, $i = 1,2,...,m$, and

g, h.

Step 2 Calculate c_j / a_j, $j = 1,2,\ldots,n$.

Step 3 If $c_1 / a_1 \geq g$, the optimal solution is as follows.

$x_j = 0, \forall j$

$v_i = 0, \forall i$

$u_i = b_i, \forall i$

$$f^*_{SKPDRCR} = g\sum_{i=1}^{m} p_i b_i$$

And let $x^*_{SKPDRCR} = x$, $u^*_{SKPDRCR} = u$, and $v^*_{SKPDRCR} = v$.

Otherwise, go to step 4.

Step 4 If $c_n / a_n \le g\sum_{i\in I_1} p_i - h\sum_{i\in I_2} p_i$, the optimal solution is as follows.

$$x_j = t_j,\ \forall j$$

$$v_i = \sum_{j=1}^{n} a_j t_j - b_i \text{ and } u_i = 0,\ \ i \in I_2$$

$$u_i = b_i - \sum_{j=1}^{n} a_j t_j \text{ and } v_i = 0,\ i \in I_1$$

$$f^*_{SKPDRCR} = \sum_{j=1}^{n} c_j t_j + g\sum_{i\in I_1} p_i (b_i - \sum_{j=1}^{n} a_j t_j) + h\sum_{i\in I_2} p_i (-b_i + \sum_{j=1}^{n} a_j t_j)$$

And let $x^*_{SKPDRCR} = x$, $u^*_{SKPDRCR} = u$, $v^*_{SKPDRCR} = v$, $k^*_{SKPDRCR} = n$ and

$b^*_{SKPDRCR} = \sum_{j=1}^{n} a_j t_j$.

Otherwise, go to step 5.

Step 5 Let $q = \left\lceil \frac{m}{2} \right\rceil$ and then calculate *k*, *cgh*1 and *cgh*2.

If $\sum_{j=1}^{n} a_j t_j < b_q$, then $k = n$. Otherwise, calculate *k* from $\sum_{j=1}^{k-1} a_j t_j \le b_q$ and

$\sum_{j=1}^{k} a_j t_j \ge b_q$.

$$cgh1 = (-c_k / a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i) \text{ and } cgh2 = (c_k / a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$$

If (*cgh1* < 0) or (*cgh2* < 0), Let $s = 1$ and go to step 6.

Otherwise, go to step 8.

Step 6 Let $q = \left\lceil \frac{m}{2} \right\rceil + s$. And then calculate *k*, *cgh*1 and *cgh*2.

If (*cgh*1 < 0) or (*cgh*2 < 0), then go to step 7.

Otherwise, go to step 8.

Step 7 Let $q = \left\lceil \frac{m}{2} \right\rceil - s$. If *q* = 0, then go to step 11. If not calculate *k*, *cgh*1 and *cgh*2.

If (*cgh1* < 0) or (*cgh2* < 0),then let *s* = *s*+1 and repeat step 6.

Otherwise, go to step 8

Step 8 Calculate x_j.

$$x_j = t_j, \quad j = 1,...,k-1$$

$$x_k = \frac{b_q - \sum_{j=1}^{k-1} a_j t_j}{a_k}$$

$$x_j = 0, \quad j = k+1,...,n$$

Step 9 Calculate u_i and v_i.

$$u_i = 0 \text{ and } v_i = b_q - b_i \quad , \ i = 1,2,...,q-1$$

$$u_q = 0 \text{ and } v_q = 0$$

$$u_i = b_i - b_q \text{ and } v_i = 0 \quad , \ i = q+1, q+2,...,m$$

Step 10 Calculate $f^*_{SKPDRCR}$.

$$f^*_{SKPDRCR} = \sum_{j=1}^{k-1} c_j t_j + c_k((b_q - \sum_{j=1}^{k-1} a_j t_j)/a_k) + h\sum_{i=1}^{q-1} p_i(b_q - b_i) + g\sum_{i=q+1}^{m} p_i(b_i - b_q)$$

And let $x^*_{SKPDRCR} = x$, $u^*_{SKPDRCR} = u$, $v^*_{SKPDRCR} = v$, $k^*_{SKPDRCR} = k$ and

$b^*_{SKPDRCR} = b_q$.

Step 11 Let $k = 1$ and $q = 1$ and go to step 12.

Step 12 Calculate $\sum_{j=1}^{k} a_j t_j$ and go to step 13.

Step 13 If $\sum_{j=1}^{k} a_j t_j > b_q$ and $\sum_{j=1}^{k} a_j t_j < b_{q+1}$, then calculate

$cgh = (c_{k+1}/a_{k+1} + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$ and go to step 14.

Otherwise, $q = q+1$ and repeat step 13.

Step 14 If $cgh \geq 0$, then go to step 15

If not, $k = k+1$ and go to step 12.

Step 15 Calculate x_j.

$$x_j = t_j, \quad j = 1,...,k$$

$$x_j = 0, \quad j = k+1,...,n$$

Step 16 Calculate u_i and v_i.

$$u_i = 0 \text{ and } v_i = \sum_{j=1}^{k} a_j t_j - b_i \quad , \ i = 1,2,...,q$$

$$u_i = b_i - \sum_{j=1}^{k} a_j t_j \text{ and } v_i = 0 \quad , \ i = q+1, q+2,...,m$$

Step 17 Calculate $f^*_{SKPDRCR}$.

$$f^*_{SKPDRCR} = \sum_{j=1}^{k} c_j t_j + h\sum_{i=1}^{q} p_i(\sum_{j=1}^{k} a_j t_j - b_i) + g\sum_{i=q+1}^{m} p_i(b_i - \sum_{j=1}^{k} a_j t_j)$$

And let $x^*_{SKPDRCR} = x$, $u^*_{SKPDRCR} = u$, $v^*_{SKPDRCR} = v$, $k^*_{SKPDRCR} = k$ and

$b^*_{SKPDRCR} = \sum_{j=1}^{k} a_j t_j$.

For the algorithm 2 of SKPDRCR, it can be generalized step by step as follows.

Step 1 Input n, m,

$c_j, a_j, t_j, \quad j = 1,2,...,n$,

$b_i, p_i, \quad i = 1,2,...,m,$ and

$g, h.$

Step2 Calculate $c'_j = \dfrac{c_j}{a_j} \quad j = 1,2,\ldots,n$

and $t'_j = a_j t_j$.

Step 3 Find all extreme points.

$$sumtp_d = \sum_{j=1}^{d} t'_j \quad , d = 1,...,n$$

$$ep_d = \sum_{j=1}^{d} t'_j \quad , d = 1,...,n$$

$$ep_d = b_i \quad , d = n+1,...,n+m$$

$$ep_d = 0 \quad , d = n+m+1$$

Sort extreme points in the increasing order.

$$eps = \text{sort}(ep)$$

Step 4 $r = 1$

Step 5 $r = r + 1$

Check that eps_r is equal to any element in $sumtp$ or b.

If eps_r is equal to any element in $sumtp$, then go to step 6.

Otherwise go to step 7.

Step 6 Check that eps_r is equal to which element in $sumtp$.

Let $eps_r = sumtp_{kk}$.

Find the first element from eps_{r+1} to eps_{n+m+1} that is also in b.

Let that element is equal to b_{iii}.

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kk} - g + (g+h)\sum_{i=1}^{iii-1} p_i$$

And go to step 8.

Step 7 Check that eps_r is equal to which element in b.

Let $eps_r = b_{ii}$.

Find the first element from eps_{r+1} to eps_{n+m+1} that is also in $sumtp$.

Let that element is equal to $\sum_{j=1}^{kkk} t'_j$.

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kkk} - g + (g+h)\sum_{i=1}^{ii-1} p_i$$

And go to step 8.

Step 8 If $SH_{(eps_{r-1}<\theta<eps_r)} > 0$, then $\theta^* = eps_{r-1}$ and go to step 9.

Otherwise, go to step 5.

Step 9 Find k^*.

$k^* = k$ such that $sumtp_{k-1} \le \theta^*$ and $sumtp_k > \theta^*$

And go to step 10.

Step 10 Calculate x_j.

$$x_j = t_j, \quad j = 1,...,k^* - 1$$

$$x_{k^*} = \frac{\theta^* - sumtp_{k^*-1}}{a_{k^*}}$$

$$x_j = 0, \quad j = k^* + 1, ..., n$$

And go to step 11.

Step 11 Calculate u_i and v_i.

$$u_i = 0 \text{ and } v_i = \theta^* - b_i, \quad \text{for } i \text{ such that } b_i < \theta^*$$

$$u_i = b_i - \theta^* \text{ and } v_i = 0, \quad \text{for } i \text{ such that } b_i \geq \theta^*$$

And go to step 12.

Step 12 Calculate $f^*_{SKPDRCR}$.

$$f^*_{SKPDRCR} = \sum_{j=1}^{n} c_j x_j + g\sum_{i=1}^{m} p_i u_i + h\sum_{i=1}^{m} p_i v_i$$

And $x^*_{SKPDRCR} = x$, $u^*_{SKPDRCR} = u$, $v^*_{SKPDRCR} = v$, $k^*_{SKPDRCR} = k^*$ and $b^*_{SKPDRCR} = \theta^*$.

2. Relaxed Problem of Stochastic Knapsack Problem with Continuous Random Capacity (SKPCRCR)

According to (2), if integer constraint is relaxed, then the relaxed problem is SKPCRCR that can be stated as follows.

(5)

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + g\int_L^U p(b)u(b)db + h\int_L^U p(b)v(b)db$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u(b) - v(b) = b$$

$$0 \leq x_j \leq t_j$$

$$\int_L^U p(b)db = 1$$

where x_j is the decision variable, for $j = 1, 2, ..., n$,

t_j is the upper bound of x_j, for $j = 1, 2, ..., n$,

$u(b)$ is the slack variable and $u(b) \geq 0$, for $L \leq b \leq U$,

$v(b)$ is the surplus variable and $v(b) \geq 0$, for $L \leq b \leq U$,

a_j is the weight coefficient of item j and $a_j \geq 0$, for $j = 1,2,...,n$,

c_j is the cost coefficient of item j and $c_j \geq 0$, for $j = 1,2,...,n$,

b is the capacity,

$p(b)$ is the probability of having capacity b, $p(b) \geq 0$, for $L \leq b \leq U$,

g is the per unit cost of having $u(b)$, $g \geq 0$, and

h is the per unit cost of having $v(b)$, $h \geq 0$.

Similarly to the discrete case, two algorithms were proposed for solving SKPCRCR. The following assumption was assumed.

$c_j / a_j \leq c_{j+1} / a_{j+1}$, for $j = 1,2,...,n-1$

2.1 Algorithm 1 of SKPCRCR

There exist three optimality conditions that are following.

Case I: the optimal solution is as follows.

$$x_j = 0, \ \forall j$$

$$v(b) = 0, \ L \leq b \leq U$$

$$u(b) = b, \ L \leq b \leq U$$

where $c_1 / a_1 \geq g$.

Proof

$$u(b) = b - \sum_{j=1}^{n} a_j x_j + v(b), \ L \leq b \leq U$$

Next part is to substitute these basic variables into the objective function as follows.

$$f = \sum_{j=1}^{n} c_j x_j + g\int_L^U p(b)(b - \sum_{j=1}^{n} a_j x_j + v(b))db + h\int_L^U p(b)v(b)db$$

$$= g\int_L^U p(b)bdb + (g+h)\int_L^U p(b)v(b)db + \sum_{j=1}^{n}(c_j - ga_j\int_L^U p(b)db)x_j$$

$$= g\int_L^U p(b)bdb + (g+h)\int_L^U p(b)v(b)db + \sum_{j=1}^{n}(c_j - ga_j)x_j$$

Since, the minimize f can be found when all reduced costs of nonbasic variables are greater than or equal to zero. In this case, nonbasic variables are x_j, for all j and v_i, for all i. In order to obtain the minimum value of f, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $(g+h)\int_L^U p(b)v(b)db$ and $(c_j - ga_j) \geq 0$, for all j.

Since $g, h \geq 0$ and $p(b) \geq 0$, for $L \leq b \leq U$, $(g+h)\int_L^U p(b)v(b)db$ is greater than or equal to zero. To minimize f, $(c_j - ga_j) \geq 0$, for all j, must be greater than or equal to zero that is equivalent to $c_j / a_j \geq g$, for all j. According to the assumption $c_j / a_j \leq c_{j+1} / a_{j+1}$, for $j = 1, 2, ..., n-1$, the condition for this case is $c_1 / a_1 \geq g$. In this case, the minimum value of f is $g\int_L^U p(b)bdb$.

Case II: the optimal solution is as follows.

$$x_j = t_j, \ \forall j$$

$$v(b) = \sum_{j=1}^{n} a_j t_j - b \text{ and } u(b) = 0, \quad b \in B_2$$

$$u(b) = b - \sum_{j=1}^{n} a_j t_j \text{ and } v(b) = 0, \quad b \in B_1$$

where $B_1 = \{L \leq b \leq U : b \geq \sum_{j=1}^{n} a_j t_j\}$, $B_2 = \{L \leq b \leq U : b < \sum_{j=1}^{n} a_j t_j\}$, and $c_n / a_n \leq g \int_{b \in B_1} p(b)db - h \int_{b \in B_2} p(b)db$.

Proof

$$x_j = t_j - r_j, \ \forall j$$

$$r_j \geq 0, \ \forall j$$

$$u(b) = b - \sum_{j=1}^{n} a_j t_j + \sum_{j=1}^{n} a_j r_j + v(b), \ b \in B_1$$

$$v(b) = -b + \sum_{j=1}^{n} a_j t_j - \sum_{j=1}^{n} a_j r_j + u(b), \quad b \in B_2$$

Next part is to substitute these basic variables into the objective function as follows.

$$
\begin{aligned}
f &= \sum_{j=1}^{n} c_j (t_j - r_j) + g \int_{b \in B_1} p(b)(b - \sum_{j=1}^{n} a_j t_j + \sum_{j=1}^{n} a_j r_j + v(b))db \\
&\quad + h \int_{b \in B_1} p(b)v(b)db + h \int_{b \in B_2} p(b)(-b + \sum_{j=1}^{n} a_j t_j - \sum_{j=1}^{n} a_j r_j + u(b))db \\
&\quad + g \int_{b \in B_2} p(b)u(b)db \\
&= \sum_{j=1}^{n} c_j t_j + g \int_{b \in B_1} p(b)(b - \sum_{j=1}^{n} a_j t_j)db + h \int_{b \in B_2} p(b)(-b + \sum_{j=1}^{n} a_j t_j)db \\
&\quad + (g+h) \int_{b \in B_1} p(b)v(b)db + (g+h) \int_{b \in B_2} p(b)u(b)db \\
&\quad + \sum_{j=1}^{n} (-c_j + g a_j \int_{b \in B_1} p(b)db - h a_j \int_{b \in B_2} p(b)db) r_j
\end{aligned}
$$

In this case, nonbasic variables are $v(b)$, for $b \in B_1$, $u(b)$, for $b \in B_2$ and r_j, for all j. In order to obtain the minimum value of *f*, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $(g+h) \int_{b \in B_1} p(b)\, v(b)\, db \geq 0$, $(g+h) \int_{b \in B_2} p(b)\, v(b)\, db \geq 0$ and $(-c_j + g a_j \int_{b \in B_1} p(b)\, db - h a_j \int_{b \in B_2} p(b)db)$, for all j.

Since $g, h \geq 0$ and $p(b) \geq 0$, for $L \leq b \leq U$, $(g+h) \int_{b \in B_1} p(b)\, v(b)\, db$ and $(g+h) \int_{b \in B_2} p(b)\, v(b)\, db$ are greater than or equal to zero. To minimize *f*, $(-c_j + g a_j \int_{b \in B_1} p(b)db - h a_j \int_{b \in B_2} p(b)db)$, for all j must be greater than or equal to zero that is equivalent to $c_j / a_j \leq g \int_{b \in B_1} p(b)db - h \int_{b \in B_2} p(b)db$, for all j. According to the assumption $c_j / a_j \leq c_{j+1} / a_{j+1}$, for $j = 1, 2, ..., n-1$, the condition for this case is

$c_n / a_n \le g \int_{b\in B_1} p(b)db - h \int_{b\in B_2} p(b)db$. In this case, the minimum value of f is $\sum_{j=1}^{n} c_j t_j$

$+g \int_{b\in B_1} p(b)(b - \sum_{j=1}^{n} a_j t_j)db + h \int_{b\in B_2} p(b)(-b + \sum_{j=1}^{n} a_j t_j)db$.

Case III: the optimal solution is as follows.

$$x_j = t_j, \quad j = 1,...,k-1$$

$$x_k = \frac{B - \sum_{j=1}^{k-1} a_j t_j}{a_k}$$

$$x_j = 0, \quad j = k+1,...,n$$

$$v(b) = B - b \text{ and } u(b) = 0, \qquad L \le b < B$$

$$u(b) = 0 \text{ and } v(b) = 0, \qquad b = B$$

$$u(b) = b - B \text{ and } v(b) = 0, \qquad B < b \le U$$

where $B \in [L,U]$ and $k \in \{1,...,n\}$ such that

$(c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db) = 0$, $\sum_{j=1}^{k-1} a_j t_j \le B$, $\sum_{j=1}^{k} a_j t_j \ge B$. If $\sum_{j=1}^{n} a_j t_j < B$, then $k = n$.

Proof

Suppose k and B have been specified.

$$f(B) = \sum_{j=1}^{k-1} c_j x_j + c_k x_k + \sum_{j=k+1}^{n} c_j x_j + g\int_L^B p(b)u(b)db + h\int_L^B p(b)u(b)db$$

$$+g\int_B^U p(b)u(b)db + h\int_B^U p(b)u(b)db \qquad (5.1)$$

Let,

$$x_j = t_j, \ j = 1,...,k-1,$$

$$x_k = (B - u(B) + v(B) - \sum_{j=1}^{k-1} a_j x_j - \sum_{j=k+1}^{n} a_j x_j) / a_k$$

$$= (B - u(B) + v(B) - \sum_{j=1}^{k-1} a_j t_j - \sum_{j=k+1}^{n} a_j x_j) / a_k,$$

$$v(b) = \sum_{j=1}^{k-1} a_j x_j + a_k x_k + \sum_{j=k+1}^{n} a_j x_j + u(b) - b$$

$$= \sum_{j=1}^{k-1} a_j x_j + a_k((B - u(B) + v(B) - \sum_{j=1}^{k\text{-}1} a_j x_j - \sum_{j=k+1}^{n} a_j x_j)/a_k)$$

$$+ \sum_{j=k+1}^{n} a_j x_j + u(b) - b$$

$$= B - b + u(b) - u(B) + v(B),\ L \le b < B\ , \text{ and}$$

$$u(b) = b + v(b) - \sum_{j=1}^{k-1} a_j x_j - a_k x_k - \sum_{j=k+1}^{n} a_j x_j$$

$$= b + v(b) - \sum_{j=1}^{k-1} a_j x_j - a_k((B - u(B) + v(B) - \sum_{j=1}^{k\text{-}1} a_j x_j - \sum_{j=k+1}^{n} a_j x_j)/a_k)$$

$$- \sum_{j=k+1}^{n} a_j x_j$$

$$= b - B + v(b) + u(B) - v(B), \quad B < b \le U\ .$$

The next part is to substitute these basic variables into equation (5.1). Thus equation (5.1) becomes as follows.

$$f(B) = \sum_{j=1}^{k-1} c_j t_j + c_k((B - u(B) + v(B) - \sum_{j=1}^{k\text{-}1} a_j t_j - \sum_{j=k+1}^{n} a_j x_j)/a_k) + \sum_{j=k+1}^{n} c_j x_j$$

$$+ g\int_L^B p(b)u(b)db + h\int_L^B p(b)(B - b + u(b) - u(B) + v(B))db$$

$$+ g\int_B^U p(b)(b - B + v(b) + u(B) - v(B))db + h\int_B^U p(b)v(b)db$$

$$= \sum_{j=1}^{k-1} c_j t_j + c_k((B - \sum_{j=1}^{k-1} a_j t_j)/a_k) + h\int_L^B p(b)(B - b)db$$

$$+ g\int_B^U p(b)(b - B)db + \sum_{j=k+1}^{n} (c_j - c_k a_j / a_k)x_j$$

$$+ (g + h)\int_L^B p(b)u(b)db + (g + h)\int_B^U p(b)v(b)db$$

$$+ (-c_k / a_k + g\int_B^U p(b)db - h\int_L^B p(b)db)u(B)$$

$$+ (c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db)v(B)$$

In this case, nonbasic variables are x_j, for $j = k+1, k+2, ..., n$, $u(b)$, for $L \le b < B$, $v(b)$, for $B < b \le U$, $u(B)$ and $v(B)$. In order to obtain the minimum value of *f*, the reduced costs of these nonbasic variables must be greater than or equal to zero that are $c_j - c_k a_j / a_k \ge 0$, for $j = k+1, k+2, ..., n$, $(g+h)\int_L^B p(b)u(b)db \ge 0$, $(g+h)\int_B^U p(b)v(b)db \ge 0$, $(-c_k/a_k + g\int_B^U p(b)db - h\int_L^B p(b)db) \ge 0$ and $(c_k/a_k + h\int_L^B p(b)db - g\int_B^U p(b)db) \ge 0$.

From assumption, $c_j / a_j \le c_{j+1} / a_{j+1}$, for $j = 1, 2, ..., n-1$ then $c_j - c_k a_j / a_k \ge 0$, for $j = k+1, k+2, ..., n$. Since $g, h \ge 0$ and $p(b) \ge 0$, for $L \le b \le U$, $(g+h)\int_L^B p(b)u(b)db$ and $(g+h)\int_B^U p(b)v(b)db$ are greater than or equal to zero. To minimize *f*, $(-c_k/a_k + g\int_B^U p(b)db - h\int_L^B p(b)db)$ and $(c_k/a_k + h\int_L^B p(b)db - g\int_B^U p(b)db)$ must be greater than or equal to zero that is equivalent to $(c_k/a_k + h\int_L^B p(b)db - g\int_B^U p(b)db) = 0$. Since $0 \le x_k \le t_k$, it is equivalent to $0 \le \frac{B - \sum_{j=1}^{k-1} a_j t_j}{a_k} \le t_k$. It is also equivalent to $\sum_{j=1}^{k-1} a_j t_j \le B$ and $\sum_{j=1}^{k} a_j t_j \ge B$. If $\sum_{j=1}^{n} a_j t_j < B$, then $k = n$.

Therefore, we select B and k such that $(c_k/a_k + h\int_L^B p(b)db - g\int_B^U p(b)db) = 0$, $\sum_{j=1}^{k-1} a_j t_j \le B$ and $\sum_{j=1}^{k} a_j t_j \ge B$. If $\sum_{j=1}^{n} a_j t_j < B$, then $k = n$. In this case, the minimum value of *f* is $\sum_{j=1}^{k-1} c_j t_j + c_k((B - \sum_{j=1}^{k-1} a_j t_j)/a_k) + h\int_L^B p(b)(B-b)db + g\int_B^U p(b)(b-B)db$.

2.2 Algorithm 2 of SKPCRCR

This algorithm was used to find the optimal solution of the SKPCRCR by using adaptations of the graphical method from the discrete case. The relaxed problem (5) can be written as follows.

(6)

$$\underset{y,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c'_j y_j + g\int_L^U p(b)u(b)db + h\int_L^U p(b)v(b)db \tag{6.1}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} y_j + u(b) - v(b) = b \tag{6.2}$$

$$0 \le y_j \le t'_j \tag{6.3}$$

$$\int_L^U p(b)db = 1 \tag{6.4}$$

where $y_j = a_j x_j$,

$c'_j = \dfrac{c_j}{a_j}$, and

$t'_j = a_j t_j$.

Objective function (6.1) can be separated into two parts as follows.

$$\underset{\theta}{\text{Minimize}}\ f = H_1(\theta) + H_2(\theta)$$

where $H_1(\theta) = \sum_{j=1}^{n} c'_j y_j$ and $H_2(\theta) = g\int_L^U p(b)u(b)db + h\int_L^U p(b)v(b)db$

Graphs of $H_1(\theta)$ and $H_2(\theta)$ of SKPCRCR were shown in Figures 3 and 4, respectively.

Let $\sum_{j=1}^{jj} y_j = \theta_{jj}$, $jj = 1, 2, ..., n$.

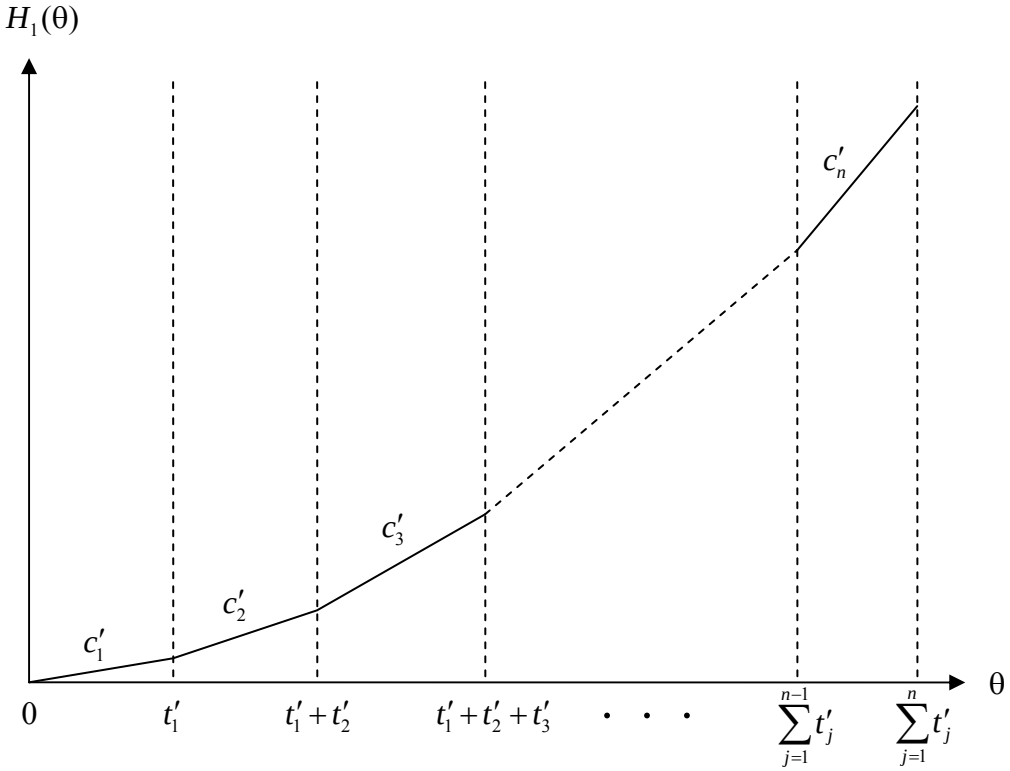

Figure 3 Graph of $H_1(\theta)$ of SKPCRCR

Let $SH1_{(t'_{j-1}\le\theta\le t'_j)} = \text{slope of graph } H_1(\theta),\ t'_{j-1} \le \theta \le t'_j$.

$= c'_j$

Since $c'_j \le c'_{j+1}$, for $j = 1,...n-1$, $H_1(\theta)$ is a piecewise-linear convex function.

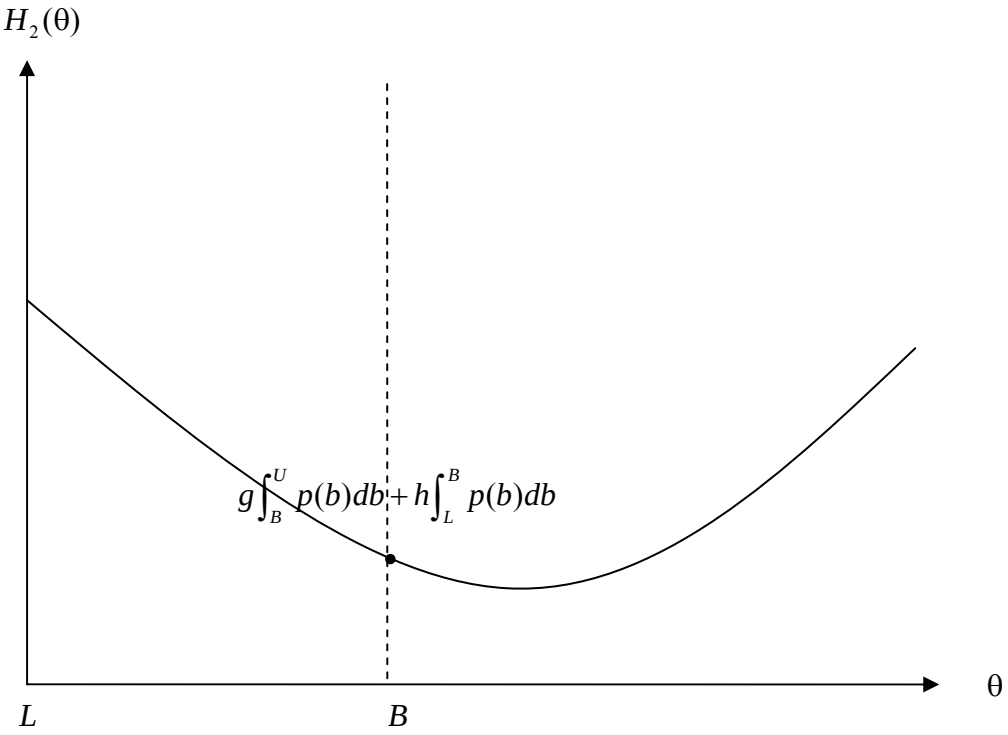

Figure 4 Graph of $H_2(\theta)$ of SKPCRCR

Let $SH2_{(\theta=B)}$ = slope of graph $H_2(\theta)$, $\theta = B$.

For $L \le \theta \le U$,

$v(b) = \theta - b$, $L \le b < \theta$,

$u(b) = b - \theta$, $\theta < b \le U$,

$v(b) = 0$, $\theta \le b \le U$, and

$u(b) = 0$, $L \le b \le \theta$.

$$H_2(\theta) = \int_\theta^U gp(b)(b-\theta)db + \int_L^\theta hp(b)(\theta - b)db$$

$$SH2_{(L\le\theta\le U)} = -g\int_\theta^U p(b)db + h\int_L^\theta p(b)db$$

$$= -g + (g+h)\int_L^\theta p(b)db$$

Since $SH2_{(\theta)} \le SH2_{(\theta+\Delta B)}$, for $L \le \theta \le U - \Delta B$, $H_2(\theta)$ is a convex function.

Since $H_1(\theta)$ is a piecewise-linear convex function and $H_2(\theta)$ is a convex function, $H_1(\theta)+H_2(\theta)$ is also a convex function.

Let $SH_\theta =$ slope of graph $H_1(\theta)+$ slope of graph $H_2(\theta)$.

$$= c'_j - g\int_\theta^U p(b)db + h\int_L^\theta p(b)db$$

The optimal solution can be found by searching for the point that has $SH_\theta = 0$. This is the same criteria as in algorithm 1 of SKPCRCR.

In this study, the algorithm 1 of SKPCRCR was written in MATLAB software as M-file program. For the algorithm 1 of SKPCRCR, it can be generalized step by step as follows.

Step 1 Input n, m,

c_j, a_j, t_j, $j = 1,2,...,n$,

$b, p(b)$, $L \le b \le U$ (conform to the distribution function of b),

g, h, and

ΔB.

Step 2 Calculate c_j / a_j, $j = 1,2,\ldots,n$.

Step 3 If $c_1 / a_1 \ge g$, the optimal solution is as follows.

$x_j = 0$, $\forall j$

$v(b) = 0$, $L \le b \le U$

$u(b) = b$, $L \le b \le U$

$f = g\int_L^U p(b)bdb$

And stop.

Otherwise, go to step 4.

Step 4 If $c_n / a_n \le g \int_{b\in B_1} p(b)db - h \int_{b\in B_2} p(b)db$, the optimal solution is as follows.

$x_j = t_j$, $\forall j$

$$v(b) = \sum_{j=1}^{n} a_j t_j - b \text{ and } u(b) = 0, \quad b \in B_2$$

$$u(b) = b - \sum_{j=1}^{n} a_j t_j \text{ and } v(b) = 0, \quad b \in B_1$$

$$f = \sum_{j=1}^{n} c_j t_j + g \int_{b \in B_1} p(b)(b - \sum_{j=1}^{n} a_j t_j) db + h \int_{b \in B_2} p(b)(-b + \sum_{j=1}^{n} a_j t_j) db$$

Otherwise, go to step 5.

Step 5 Let $s = 1$.

Step 6 Let $B = s(\Delta B)$ and then calculate *k* and *cgh*.

If $\sum_{j=1}^{n} a_j t_j < B$, then $k = n$. Otherwise, calculate *k* from $\sum_{j=1}^{k-1} a_j t_j \le B$ and

$\sum_{j=1}^{k} a_j t_j \ge B$.

$cgh = (c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db)$

If ($cgh < 0$), let $s = s + 1$ and repeat step 6.

Otherwise, go to step 7.

Step 7 Let $B_1 = B - \Delta B$ and $B_2 = B$ and calculate $k_1, k_2, f(B_1)$ and $f(B_2)$.

If $f(B_1) < f(B_2)$, then $f^*_{SKPCRCR} = f(B_1), k^*_{SKPCRCR} = k_1$ and $B^*_{SKPCRCR} = B_1$.

Otherwise, $f^*_{SKPCRCR} = f(B_2), k^*_{SKPCRCR} = k_2$ and $B^*_{SKPCRCR} = B_2$.

Step 8 Calculate $x_j, u(b)$ and $v(b)$.

$$x_j = t_j, \quad j = 1, ..., k^*_{SKPCRCR} - 1$$

$$x_{k^*_{SKPCRCR}} = \frac{B - \sum_{j=1}^{k^*_{SKPCRCR}-1} a_j t_j}{a_{k^*_{SKPCRCR}}}$$

$$x_j = 0, \quad j = k^*_{SKPCRCR} + 1, ..., n$$

$$v(b) = B^*_{SKPCRCR} - b \text{ and } u(b) = 0, \qquad L \le b < B^*_{SKPCRCR}$$

$$u(b) = 0 \text{ and } v(b) = 0 \qquad b = B^*_{SKPCRCR}$$

$$u(b) = b - B^*_{SKPCRCR} \text{ and } v(b) = 0, \qquad B^*_{SKPCRCR} < b \le U$$

$$x^*_{SKPCRCR} = x, u^*_{SKPCRCR} = u(b) \text{ and } v^*_{SKPCRCR} = v(b)$$

2.3 Monte Carlo Simulation

Monte Carlo simulation is the natural alternative for using in stochastic programs. Therefore, the Monte Carlo simulation is applied to solve SKPCRCR. In this approach, a numerous number of samples is generated. Then each sample is solved directly as a deterministic problem. The optimal solution is the expected value of all solutions.

For each sample, the problem is as follows.

(7)

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + gu + hv$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u - v = b$$

$$0 \le x_j \le t_j$$

For problem (7), v always zero because the objective value of solution that have $v > 0$ can be smaller by reducing the value of x_j. Therefore, the problem can be reduced to problem (8).

(8)

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + gu$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u = b$$

$$0 \le x_j \le t_j$$

In this study, the Monte Carlo simulation of SKPCRCR was written in MATLAB software as M-file program. For the Monte Carlo simulation of SKPCRCR, it can be generalized step by step as follows.

Step 1 Input n, m,

$$c_j, a_j, t_j, \qquad j = 1, 2, ..., n,$$

L, U, and
g, h.

Step 2 Calculate c_j / a_j, $j = 1,2,\ldots,n$.

Step 3 Let $tr = 1$.

Step 4 If $tr = 100,000$, then go to step 13.

Step 5 Generate b, $L \le b \le U$ (conform to the distribution function of b).

Step 6 Let $c_{n+1} = g$,

$a_{n+1} = 1$, and
$t_{n+1} = \infty$.

Step 7 Sort c_j / a_j, for $j = 1,2,...,n+1$, in ascending order referred as cs_j / as_j. Then, find cs_j, as_j and ts_j with a sorting position vector e.

Step 8 Calculate $\sum_{j=1}^{n+1} as_j ts_j$.

Step 9 If $as_1 ts_1 > b$, then $k^* = 1$.

Otherwise, find k^* from $\sum_{j=1}^{k^*+1} as_j ts_j \le b$ and $\sum_{j=1}^{k^*} as_j ts_j > b$.

Step 10 Calculate xs_j.

$$xs_j = ts_j, \text{ for } j = 1,...,k^* - 1$$
$$xs_j = (b - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$$
$$xs_j = 0, \text{ for } j = k^* + 1,...,n+1$$

Step 11 Revert xs_j to x_j, $j = 1,2,...,n$.

Step 12 Let $tr = tr + 1$ and go to step 4.

Step 13 Calculate $\overline{x_j}$, $B^*_{SKPCRCR}$ and $f^*_{SKPCRCR}$.

$$B^*_{SKPCRCR} = \sum_{j=1}^{n} a_j \overline{x_j}$$

$$f^*_{SKPCRCR} = \sum_{j=1}^{n} c_j \overline{x_j} + h \int_L^{B^*_{SKPCRCR}} p(b)(B^*_{SKPCRCR} - b)\, db$$
$$+ g \int_{B^*_{SKPCRCR}}^{U} p(b)(b - B^*_{SKPCRCR})\, db$$

3. Heuristics for solving SKPDRC and SKPCRC

The reasons for using heuristics for SKPDRC and SKPCRC are to save computing time and to provide the optimal solution or near-optimal solution.

If all elements in $x^*_{SKPDRCR}$ are integer, then the solution of SKPDRC is the same as the solution of SKPDRCR (i.e., $x^*_{SKPDRC} = x^*_{SKPDRCR}$, $u^*_{SKPDRC} = u^*_{SKPDRCR}$, $v^*_{SKPDRC} = v^*_{SKPDRCR}$, and $f^*_{SKPDRC} = f^*_{SKPDRCR}$). Similarly, if all elements in $x^*_{SKPCRCR}$ are integer, then the solution of SKPCRC is the same as the solution of SKPCRCR (i.e., $x^*_{SKPCRC} = x^*_{SKPCRCR}$, $u^*_{SKPCRC} = u^*_{SKPCRCR}$, $v^*_{SKPCRC} = v^*_{SKPCRCR}$, and $f^*_{SKPCRC} = f^*_{SKPCRCR}$). Otherwise, the solution of SKPDRC or SKPCRC can be found by using the proposed heuristics, which use the optimal capacity of the relaxed problem, i.e., $b^*_{SKPDRCR}$ for discrete capacity and $B^*_{SKPCRCR}$ for continuous capacity, as the upper bound of capacity. Due to SKPDRCR is a piecewise convex function and SKPCRCR is a convex function, a good or near optimal solution should have the weight sum ($\sum_{j=1}^{n} a_j x_j$) near optimal capacity of the relaxed problem.

The flow charts of heuristics for solving SKPDRC and SKPCRC were shown in Figures 5 and 6, respectively.

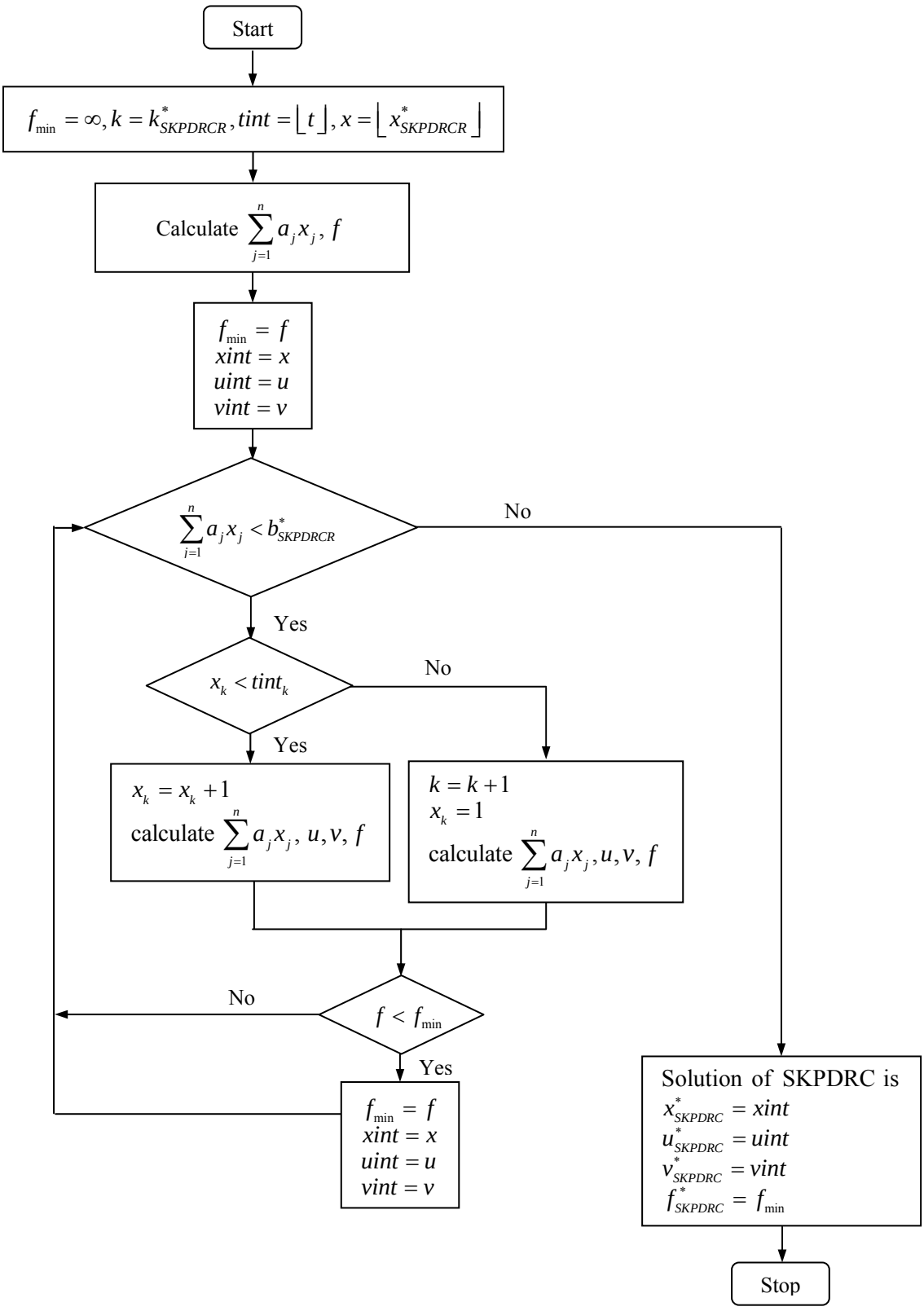

Figure 5 Heuristic for SKPDRC

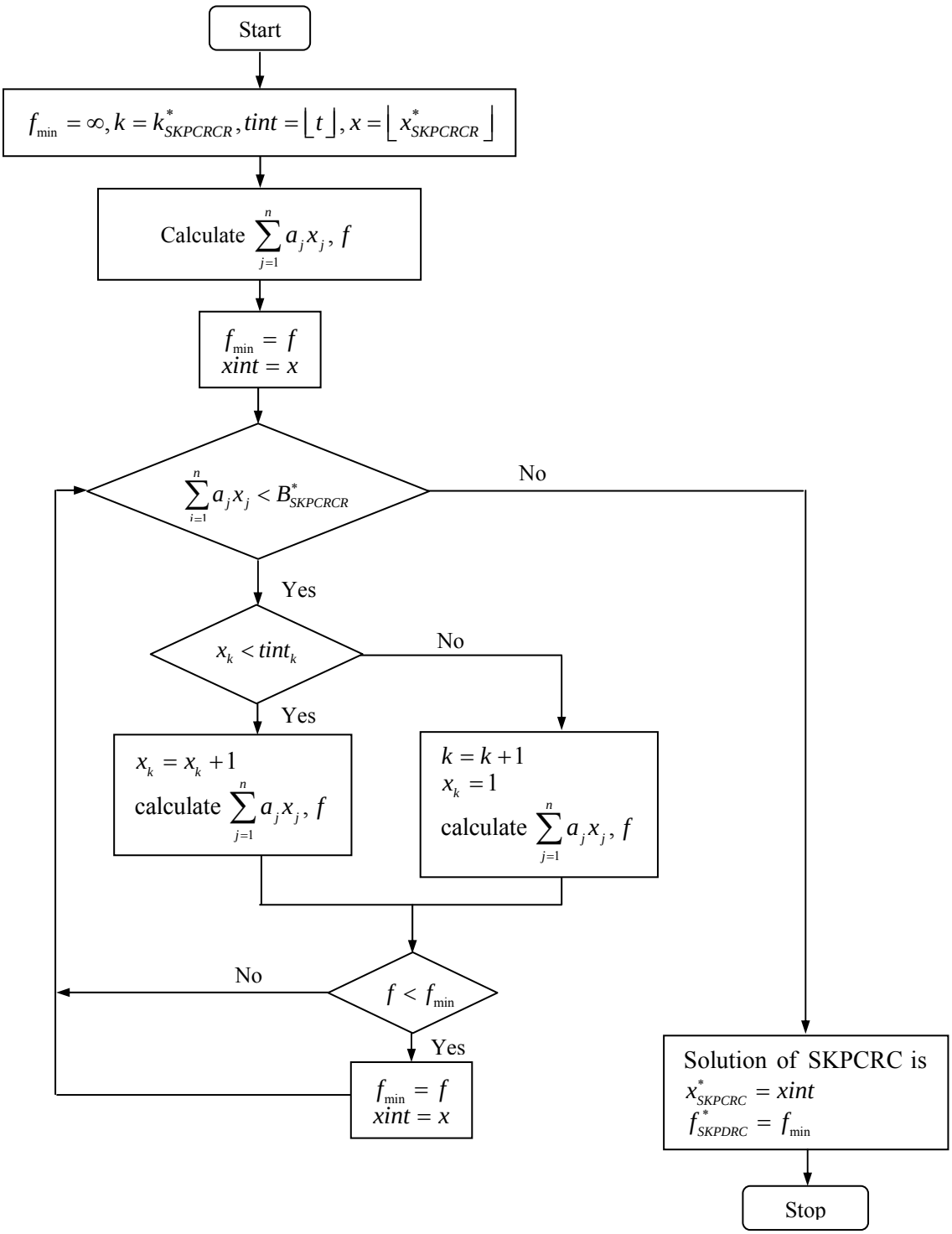

Figure 6 Heuristic for SKPCRC

From Figures 5 and 6, the heuristics start from rounding down $x^*_{SKPDRCR}$ or $x^*_{SKPCRCR}$ and the upper bound of the decision variable (*t*). Let *x* is the rounding down of $x^*_{SKPDRCR}$ or $x^*_{SKPCRCR}$, *tint* is the rounding down of *t*, *k* is the $k^*_{SKPDRCR}$ or $k^*_{SKPCRCR}$, and $f_{\min}$ is equal to infinity. Then calculate the weight sum and the objective value, and keep the minimum objective value ($f_{\min}$), the solution of the SKPDRC (*xint*, *uint*, *vint*) or the solution of the SKPCRC (*xint*). Next, go to the capacity checking point. If the weight sum is less than the optimal capacity of the relaxed problem, then go to the upper bounded checking point. Otherwise, integer solution is found.

For the upper bound checking point, if x_k is less than its upper bound $(tint_k)$, then let $x_k = x_k + 1$ and calculate the weight sum, the slack variable (*u*), the surplus variable (*v*) and the objective value (f). Otherwise, let $k = k + 1$ and $x_k = 1$, and then calculate the weight sum, the slack variable (*u*), the surplus variable (*v*) and the objective value (f).

Next, go to the objective value checking point. If the objective value is less than $f_{\min}$, then the new solution is found. Therefore, the old solution is replaced by the new solution. And then go back to the capacity checking point. Otherwise, keep the old solution and go back to the capacity checking point. And continue these steps until the weight sum is greater than or equal to the optimal capacity of the relaxed problem. Finally, the solution of SKPDRC or SKPCRC is found.

RESULTS AND DISCUSSION

In this section, the results of the experiments for the stochastic knapsack problem with discrete and continuous random capacity were presented.

1. Results for Stochastic Knapsack Problem with Discrete Random Capacity (SKPDRC)

The experiments for SKPDRCR and SKPDRC were conducted by varying *m* and *n*. The computing time (excluding parameter generating) and the solutions obtained were collected and compared. The input data were generated as follows.

c_j, a_j, t_j, for $j = 1, 2, ..., n$, were generated with uniform[0,10].

b_i, for $i = 1, 2, ..., m$, was generated with uniform$[0, \sum_{j=1}^{n} a_j t_j]$.

g, h were generated with uniform[0,10].

$p_i = \frac{pp_i}{\sum_{i=1}^{m} pp_i}$ where pp_i, for $i = 1, 2, ..., m$, was generated with uniform [0,1].

The reasons for generating c_j, a_j, t_j, g, and h via uniformly distribution [0,10] were to get the various positions of the optimal solution and also the optimal objective function value will not be too large. Moreover, the reason for generating b_i was generated with uniform $[0, \sum_{j=1}^{n} a_j t_j]$ was to ensure that all items fit in the capacity. Furthermore, the reason of generating p_i as such is the sum of all p_i must be equal to 1.

For SKPDRCR, the only computing time was compared, since the algorithms 1 and 2 of SKPDRCR and the general purpose method using CPLEX for solving SKPDRCR gave the optimal solution.

Fifty hundred generated samples were conducted to compare computing time of the algorithms 1, 2 and CPLEX for large problem size, (i.e., $m \geq 2,500$ and $n \geq 2,500$). Otherwise, a hundred generated samples were conducted.

The computing time of the algorithms 1 and 2 of SKPDRCR, and the general purpose method using CPLEX for solving SKPDRCR were presented in Tables 1, 2 and 3, respectively.

Table 1 Computing time (sec) of the algorithm 1 of SKPDRCR

n	m								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	0.011	0.029	0.056	0.085	0.114	0.315	0.901	1.734	2.704
250	0.031	0.077	0.156	0.218	0.310	0.817	2.287	3.008	5.574
500	0.081	0.207	0.373	0.557	0.761	1.963	3.864	7.868	11.778
750	0.150	0.368	0.595	1.101	1.309	2.389	5.598	7.938	13.323
1000	0.217	0.483	0.863	1.471	2.072	4.433	8.458	11.703	18.023
2,500	0.888	2.666	4.872	6.826	10.415	29.388	63.461	71.715	91.538
5,000	4.350	8.546	19.847	26.469	38.478	119.760	197.642	250.369	349.069
7,500	8.888	30.717	54.601	64.797	89.800	156.285	232.818	317.560	384.534
10,000	10.052	34.401	78.140	118.136	153.734	196.675	322.130	1,450.716	2,615.406

Table 2 Computing time (sec) of the algorithm 2 of SKPDRCR

n	m								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	0.004	0.007	0.016	0.029	0.043	0.192	0.685	1.514	2.411
250	0.008	0.012	0.022	0.035	0.051	0.203	0.694	1.528	2.427
500	0.017	0.022	0.033	0.050	0.065	0.222	0.701	1.543	2.445
750	0.029	0.037	0.050	0.066	0.085	0.241	0.713	1.570	2.478
1,000	0.044	0.051	0.069	0.087	0.113	0.297	0.767	1.762	2.519
2,500	0.199	0.203	0.230	0.247	0.309	0.605	1.231	2.014	2.849
5,000	0.652	0.666	0.687	0.756	0.894	1.219	2.193	3.129	4.374
7,500	1.394	1.404	1.461	1.545	1.633	2.108	3.063	4.263	5.682
10,000	2.475	2.492	2.532	2.632	2.792	3.330	4.732	5.935	8.520

Table 3 Computing time (sec) of the general purpose method using CPLEX for solving SKPDRCR

n	m								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	1.656	1.726	1.809	1.873	2.075	2.390	4.654	7.049	12.743
250	1.696	1.808	1.885	2.058	2.327	3.341	8.012	12.585	18.468
500	1.733	1.899	2.015	2.346	2.705	5.176	11.079	17.768	23.658
750	1.786	1.961	2.266	2.691	3.270	6.028	13.361	20.967	36.571
1,000	1.977	2.023	2.487	3.047	3.694	7.259	15.133	31.628	63.481
2,500	2.131	2.642	4.193	5.545	7.581	15.876	210.443	2,365.781	N/A
5,000	2.665	4.074	6.681	10.019	12.954	222.091	N/A	N/A	N/A
7,500	3.130	5.387	9.438	14.794	22.194	2,696.544	N/A	N/A	N/A
10,000	3.959	6.745	12.776	24.423	59.356	N/A	N/A	N/A	N/A

N/A = solution of CPLEX cannot be found because of out of memory with PC Intel Pentium M Processor 1.6 GHz., 512 MB RAM.

According to Tables 1, 2 and 3, the computing time increased when m and n were increased. The computing time of both the algorithm 1 and the general purpose method using CPLEX take extremely long when m and n were large. Moreover, CPLEX cannot solve SKPDRCR of large m and n (i.e., $n = 2,500$ where $m \geq 10,000$, $n = 5,000$ where $m \geq 5,000$, $n = 7,500$ where $m \geq 5,000$ and $n = 10,000$ where $m \geq 2,500$). The best algorithm for solving SKPDRCR was presented in Table 4.

Table 4 The best algorithm for solving SKPDRCR (Algorithm1 (I), Algorithm2 (II) and CPLEX (III))

n	*m*								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	II I(0.01) III(1.65)	II I(0.02) III(1.72)	II I(0.04) III(1.79)	II I(0.06) III(1.84)	II I(0.07) III(2.03)	II I(0.12) III(2.20)	II I(0.22) III(3.97)	II I(0.22) III(5.54)	II I(10.33) III(10.33)
250	II I(0.02) III(1.68)	II I(0.06) III(1.80)	II I(0.13) III(1.86)	II I(0.18) III(2.02)	II I(0.26) III(2.28)	II I(0.61) III(3.14)	II I(1.59) III(7.32)	II I(1.48) III(11.06)	II I(3.15) III(16.04)
500	II I(0.06) III(1.72)	II I(1.18) III(1.88)	II I(0.34) III(1.98)	II I(0.51) III(2.30)	II I(0.70) III(2.64)	II I(1.74) III(4.95)	II I(3.16) III(10.38)	II I(6.32) III(16.22)	II I(9.33) III(21.21)
750	II I(0.12) III(1.76)	II I(0.33) III(1.92)	II I(0.54) III(2.22)	II I(1.03) III(2.62)	II I(1.22) III(3.19)	II I(2.15) III(5.79)	II I(4.89) III(12.65)	II I(6.37) III(19.40)	II I(10.84) III(34.09)
1,000	II I(0.17) III(1.93)	II I(0.43) III(1.97)	II I(0.79) III(2.42)	II I(1.38) III(2.96)	II I(1.96) III(3.58)	II I(4.14) III(6.96)	II I(7.69) III(14.37)	II I(9.94) III(29.87)	II I(15.50) III(60.96)
2,500	II I(0.69) III(1.93)	II III(2.44) I(2.46)	II III(3.96) I(4.64)	II III(5.30) I(6.58)	II III(7.27) I(10.11)	II III(15.27) I(28.78)	II I(62.23) III(209.21)	II I(67.70) III(2,363.77)	II I(88.69) III(N/A)
5,000	II III(2.01) I(3.70)	II III(3.41) I(7.88)	II III(5.99) I(19.16)	II III(9.26) I(25.71)	II III(12.06) I(37.58)	II I(118.54) III(220.87)	II I(195.45) III(N/A)	II I(247.24) III(N/A)	II I(344.69) III(N/A)
7,500	II III(1.74) I(7.49)	II III(3.98) I(29.31)	II III(7.98) I(53.14)	II III(13.25) I(63.25)	II III(20.56) I(88.17)	II I(154.18) III(2,694.44)	II I(229.75) III(N/A)	II I(313.30) III(N/A)	II I(378.85) III(N/A)
10,000	II III(1.48) I(7.58)	II III(4.25) I(31.91)	II III(10.24) I(75.61)	II III(21.79) I(115.50)	II III(56.56) I(150.94)	II I(193.34) III(N/A)	II I(317.40) III(N/A)	II I(1,444.78) III(N/A)	II I(2,606.89) III(N/A)

\# = The best algorithm

#(xxx) = The second algorithm and saving time of the best algorithm compared with this algorithm.

#(xxx) = The worst algorithm and saving time of the best algorithm compared with this algorithm.

N/A = Solution of CPLEX cannot be found because of out of memory with PC Intel Pentium M Processor 1.6 GHz., 512 MB RAM.

According to Table 4, the algorithm 2 of SKPDRCR was the best algorithm for all pairs of *m* and *n*. Furthermore, the algorithm 1 was better than the general purpose method using CPLEX when *n* was small or *n* and *m* were large. There were some cases that the general purpose method using CPLEX was better than the algorithm 1 as follows:

1. $n \geq 5,000$ and $m \leq 1,000$
2. $n = 2,500$ and $250 \leq m \leq 2,500$

For the algorithm 1 of SKPDRC, there are $2 \times m \times n + 1$ iterations for the worst case. Therefore, the worst case time complexity is $O(mn)$ that the reason of the computing time of Table 1 increased as polynomial. For the algorithm 2 of SKPDRC, there are $m + n + 1$ iterations for the worst case. Therefore, the worst case time complexity is $O(m+n)$ that the reason of the computing time of Table 2 grew as linear. The ratios of computing time of the algorithm 2 of SKPDRCR divided by computing time of the general purpose method using CPLEX for solving SKPDRCR was shown in Table 5.

Table 5 The ratios of computing time of the algorithm 2 of SKPDRCR divided by computing time of the general purpose method using CPLEX for solving SKPDRCR

n	m								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	460.028	239.694	111.673	64.378	47.924	12.442	6.794	4.657	5.286
250	217.410	147.016	86.862	59.488	45.803	16.435	11.546	8.234	7.610
500	100.191	84.790	61.048	46.926	41.933	23.356	15.811	11.512	9.674
750	60.949	53.431	45.048	40.582	38.603	25.045	18.752	13.354	14.759
1,000	45.335	39.909	35.840	35.184	32.571	24.465	19.730	17.953	25.201
2,500	10.689	13.021	18.213	22.475	24.551	26.250	171.009	1174.843	N/A
5,000	4.085	6.115	9.730	13.251	14.496	182.146	N/A	N/A	N/A
7,500	2.245	3.838	6.462	9.577	13.595	1278.953	N/A	N/A	N/A
10,000	1.599	2.707	5.046	9.280	21.259	N/A	N/A	N/A	N/A

N/A = Solution of CPLEX cannot be found because of out of memory with PC Intel Pentium M Processor 1.6 GHz., 512 MB RAM.

According to Table 5, when *n* increased, the ratios were decreased until reached the fluctuation point of *n*, and then they were increased. If *m* was small, the fluctuation point of *n* would be the high value of *n*. For example, when *m* was 100, 250, 500, or 750, the fluctuation point of *n* would be more than 7,500, which can not be seen the fluctuation trend from Table 5. On the other hand, if *m* was large, the fluctuation point of *n* would be at the small value of *n*. For example, when *m* was 2,500, 5,000, 7,500, or 10,000, the fluctuation point of *n* would be less than 250, which can not be seen the fluctuation trend from Table 5. When *m* was equal to 1,000, the fluctuation point of *n* would be the value between 5,000 and 10,000. Similarly, when *m* increased, the ratios were decreased until reached the fluctuation point of *m*, and then they were increased. If *n* was small, the fluctuation point of *m* would be the high value of *m*. For example, when *n* was 250 or 500 the fluctuation point of *m* would be more than 7,500, which can not be seen the fluctuation trend from Table 5. Remark that when *n* was equal to 100, the fluctuation point of *m* would be the value between 5,000 and 10,000. On the other hand, if *n* was large, the fluctuation point of *m* would be at the small value of *m*. For example, when *n* was 2,500, 5,000, 7,500, or 10,000, the fluctuation point of *m* would be less than 250, which can not be seen the fluctuation trend from Table 5. When *n* was equal to 750 or 1,000, the fluctuation point of *m* was between 5,000 and 10,000.

Since the algorithm 2 was better than the algorithm 1, it was selected for finding the initial solution for the heuristic of SKPDRC. Computing time of the proposed heuristic for SKPDRC where the relaxed problem was solved by using algorithm 2 of SKPDRCR was presented in Table 6.

Table 6 Computing time (sec) of the proposed heuristic for SKPDRC where the relaxed problem was solved by using the algorithm 2 of SKPDRCR

n	m								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	0.004	0.008	0.018	0.032	0.046	0.199	0.696	1.530	2.432
250	0.010	0.016	0.026	0.040	0.058	0.220	0.724	1.570	2.480
500	0.023	0.030	0.042	0.062	0.079	0.250	0.754	1.637	2.562
750	0.040	0.050	0.068	0.086	0.108	0.284	0.795	1.703	2.632
1,000	0.060	0.071	0.094	0.117	0.149	0.361	0.876	1.948	2.742
2,500	0.272	0.306	0.324	0.345	0.430	0.814	1.562	2.432	3.360
5,000	0.893	0.936	0.963	1.058	1.271	1.870	2.944	4.071	5.582
7,500	1.821	1.945	2.023	2.177	2.326	3.019	4.185	5.722	7.501
10,000	3.422	3.478	3.561	3.810	4.022	4.867	6.476	8.203	11.374

SKPDRC is the mixed integer programming (MIP) model. CPLEX uses branch and cut search procedure for solving MIP model. The branch and cut search procedure manages a search tree consisting of nodes. Every node represents an LP or QP subproblem to be processed. It will be solved and checked for integrality, and perhaps be further analyzed. Nodes are called active if they have not yet been processed. Once a node has been processed, it is no longer active. CPLEX processes active nodes in the tree until either no active node is available or some limit, e.g. MIP gap (expressed as a percentage of the incumbent solution) has been reached. Under the default settings, CPLEX will terminate the search when the MIP gap is lower than 0.0001 (0.01%). The computing time of the general purpose method using CPLEX for solving SKPDRC was presented in Table 7.

Table 7 Computing time (sec) of the general purpose method using CPLEX for solving SKPDRC

n	m								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	2.052	2.100	2.151	2.227	2.321	2.806	5.409	10.908	18.525
250	2.078	2.126	2.269	2.382	2.573	4.301	8.341	14.336	22.016
500	2.109	2.234	2.577	2.952	3.341	6.292	13.927	22.344	31.903
750	2.142	2.402	2.943	3.545	4.303	8.283	16.505	31.871	59.629
1,000	2.182	2.586	3.311	4.206	5.356	10.501	21.489	53.411	96.151
2,500	2.520	3.564	5.446	7.756	10.714	32.491	346.549	3,711.499	N/A
5,000	3.311	5.646	10.070	14.609	19.471	350.791	N/A	N/A	N/A
7,500	4.244	7.909	14.431	25.636	47.732	4,526.133	N/A	N/A	N/A
10,000	5.420	10.487	19.759	48.107	95.170	N/A	N/A	N/A	N/A

N/A = Solution of CPLEX cannot be found because of out of memory with PC Intel Pentium M Processor 1.6 GHz., 512 MB RAM.

According to Tables 6 and 7, the proposed heuristic of SKPDRC had shorter computing time than the general purpose method using CPLEX. The ratios of computing time of the proposed heuristic for SKPDRC, where the relaxed problem was solved by using the algorithm 2 of SKPDRCR, divided by computing time of the general method using CPLEX for solving SKPDRC was shown in Table 8.

Table 8 The ratios of computing time of the proposed heuristic for SKPDRC, where the relaxed problem was solved by using the algorithm 2 of SKPDRCR, divided by computing time of the general purpose method using CPLEX for solving SKPDRC

n	*m*								
	100	250	500	750	1,000	2,500	5,000	7,500	10,000
100	466.386	253.000	118.203	70.259	50.236	14.137	7.775	7.128	7.617
250	201.738	137.155	87.250	59.097	44.364	19.560	11.523	9.129	8.878
500	90.498	75.486	61.364	47.457	42.504	25.139	18.480	13.649	12.453
750	53.406	48.032	43.537	41.170	39.768	29.196	20.760	18.711	22.655
1,000	36.664	36.576	35.227	35.915	35.997	29.058	24.531	27.419	35.063
2,500	9.272	11.645	16.819	22.462	24.933	39.925	221.834	1526.173	N/A
5,000	3.708	6.032	10.461	13.805	15.317	187.599	N/A	N/A	N/A
7,500	2.331	4.066	7.135	11.774	20.521	1499.216	N/A	N/A	N/A
10,000	1.584	3.015	5.549	12.626	23.664	N/A	N/A	N/A	N/A

N/A = Solution of CPLEX cannot be found because of out of memory with PC Intel Pentium M Processor 1.6 GHz., 512 MB RAM.

According to Table 8, when *n* increased, the ratios were decreased until reached the fluctuation point of *n*, and then they were increased. If *m* was small, the fluctuation point of *n* would be the high value of *n*. For example, when *m* was 100, 250, or 500, the fluctuation point of *n* would be more than 7,500, which can not be seen fluctuation trend from Table 8. On the other hand, if *m* was large, the fluctuation point of *n* would be at the small value of *n*. For example, when *m* was 2,500, 5,000, 7,500, or 10,000, the fluctuation point of *n* would be less than 250, which can not be seen the fluctuation trend from Table 8. When *m* was equal to 750, the fluctuation point of *n* was between 5,000 and 10,000, and when *m* was equal to 1,000, the fluctuation point of *n* was between 2,500 and 7,500. Similarly, when *m* increased, the ratios were decreased until they were reached the fluctuation point of *m*, and then they were increased. If *n* was small, the fluctuation point of *m* would be the high value of *m*. For example, when *n* was 250 or 500 the fluctuation point of *m* would be more than 7,500, which can not be seen the fluctuation trend from Table 8. Remark that

when *n* was equal to 100, the fluctuation point of *m* would be the value between 5,000 and 10,000. On the other hand, if *n* was large, the fluctuation point of *m* would be at the small value of *m*. For example, when *n* was 2,500, 5,000, 7,500, or 10,000, the fluctuation point of *m* would be less than 250, which can not be seen the fluctuation trend from Table 8. When *n* was equal to 750, the fluctuation point of *m* was between 5,000 and 10,000, and when *n* was equal to 1,000, the fluctuation point of *m* was between 2,500 and 7,500.

.

Fifty generated samples were conducted to compare the quality of solutions between the proposed heuristic for SKPDRC and the general purpose method using CPLEX. The objective value of the proposed heuristic for SKPDRC was compared with objective value of the general purpose method using CPLEX. The percent win/tie/lose of the objective values of the proposed heuristic for SKPDRC compared with the general purpose method using CPLEX were presented in Table 9.

Table 9 Percent Win/Tie/Lose of the objective values of the proposed heuristic for SKPDRC compared with the general purpose method using CPLEX

n	*m* = 100			*m* = 500			*m* = 1,000			*m* = 5,000		
	Win	Tie	Lose	Win	Tie	lose	Win	Tie	lose	Win	Tie	Lose
100	40	56	4	48	50	2	56	44	0	58	42	0
500	80	18	2	84	14	2	78	20	2	84	14	2
1,000	86	12	2	86	14	0	90	10	0	88	12	0
5,000	100	0	0	100	0	0	100	0	0	N/A	N/A	N/A

N/A = Solution of CPLEX cannot be found because of out of memory with PC Intel Pentium M Processor 1.6 GHz., 512 MB RAM.

According to Table 9, percent wining of the proposed heuristic for SKPDRC was more than percent losing and percent wining was significantly more than percent losing when *n* was large. However, the percent losing of the proposed heuristic for SKPDRC was 1.0667% where 90% confidence interval was [1.0496%, 2.2284%].

2. Results for Stochastic Knapsack Problem with Continuous Random Capacity (SKPCRC)

The experiments for SKPCRCR and SKPCRC were conducted by varying *m* and *n*. The computing time (excluding parameter generating) and solutions obtained were collected and compared. The input data were generated as follows.

c_j, a_j, t_j, for $j = 1, 2, ..., n$, were generated with uniform[0,10].

b was normally distributed where mean was generated with uniform$[0, \sum_{j=1}^{n} a_j t_j]$ and standard deviation was generated with uniform[0, *n*].

g, h were generated with uniform[0,10].

$$\Delta B = 0.01\sum_{j=1}^{n} a_j t_j,\ 0.001\sum_{j=1}^{n} a_j t_j,\ 0.0001\sum_{j=1}^{n} a_j t_j,\ 0.00001\sum_{j=1}^{n} a_j t_j$$

The reasons for generating c_j, a_j, t_j, g, and h via uniformly distribution [0,10] were to get the various positions of the optimal solution and the optimal objective function value will not be too large. Moreover, the reason for generating *b* was generated with normally distributed where mean was generated with uniform $[0, \sum_{j=1}^{n} a_j t_j]$ and standard deviation was generated with uniform [0, *n*] was to ensure that all items fit in the capacity. For stepping size (ΔB), we conducted the experiments with four values of ΔB in order to select the appropriate ΔB for this studied problem.

The computing time of the algorithm 1 (or algorithm 2) and the Monte Carlo simulation for solving SKPCRCR were presented in Table 10. The computing time of the proposed heuristic for SKPCRC where the relaxed problem was solved by using the algorithm 1 (or algorithm 2) and the Monte Carlo simulation of SKPCRCR were presented in Table 11.

Table 10 Computing time (sec) of the algorithm 1 (or algorithm 2) and the Monte Carlo simulation for solving SKPCRCR

n	Algorithm 1 (or Algorithm 2)				Monte Carlo Simulation
	ΔB				
	$0.01\sum_{j=1}^{n} a_j t_j$	$0.001\sum_{j=1}^{n} a_j t_j$	$0.0001\sum_{j=1}^{n} a_j t_j$	$0.00001\sum_{j=1}^{n} a_j t_j$	
100	0.0186 (100%) [–]	0.1284 (100%) [–]	1.1362 (100%) [–]	11.0618 (28%) [17.6, 38.4]	4.6029
250	0.0292 (100%) [–]	0.2557 (100%) [–]	2.3814 (100%) [–]	23.4010 (26%) [15.8, 36.2]	7.0102
500	0.0665 (100%) [–]	0.5349 (100%) [–]	5.1265 (96%) [91.4, 100.0]	50.5982 (24%) [14.1, 33.9]	11.0761
750	0.0932 (100%) [–]	0.7895 (100%) [–]	7.5840 (88%) [80.4, 95.6]	75.0517 (22%) [12.4, 31.6]	15.4615
1,000	0.1405 (100%) [–]	1.1420 (100%) [–]	11.0026 (80%) [70.7, 89.3]	109.1829 (20%) [10.7, 29.3]	19.4031
2,500	0.5119 (100%) [–]	4.4021 (100%) [–]	41.9293 (64%) [52.8, 75.2]	414.3567 (20%) [10.7, 29.3]	46.8583
5,000	1.8755 (100%) [–]	18.8473 (100%) [–]	155.9795 (46%) [34.4,57.6]	1,766.7708 (16%) [7.5, 24.5]	98.5195

(%xxx) = Percent winning of the algorithm 1 (or algorithm 2) compared to the Monte Carlo simulation for solving SKPCRCR in terms of computing time

[xx.x, xx.x] = 90% confidence interval of percent winning in computing time

[–] = Cannot find the confidence interval because of percent winning in computing time is 100%

Table 11 Computing time (sec) of the proposed heuristic for SKPCRC where the relaxed problem was solved by using the algorithm 1 (or algorithm 2) and the Monte Carlo simulation of SKPCRCR

n	Algorithm 1 (or Algorithm 2)				Monte Carlo Simulation
	ΔB				
	$0.01\sum_{j=1}^{n} a_j t_j$	$0.001\sum_{j=1}^{n} a_j t_j$	$0.0001\sum_{j=1}^{n} a_j t_j$	$0.00001\sum_{j=1}^{n} a_j t_j$	
100	0.0302 (100%) [–]	0.1432 (100%) [–]	1.1507 (100%) [–]	11.0762 (28%) [17.6, 38.4]	4.6147
250	0.0422 (100%) [–]	0.2768 (100%) [–]	2.4039 (100%) [–]	23.4220 (26%) [15.8, 36.2]	7.0307
500	0.0993 (100%) [–]	0.5696 (100%) [–]	5.1614 (96%) [91.4, 100.0]	50.6351 (24%) [14.1, 33.9]	11.1218
750	0.1356 (100%) [–]	0.8344 (100%) [–]	7.6287 (88%) [80.4, 95.6]	75.0950 (22%) [12.4, 31.6]	15.5035
1,000	0.2062 (100%) [–]	1.2015 (100%) [–]	11.0719 (80%) [70.7, 89.3]	109.2460 (20%) [10.7, 29.3]	19.4641
2,500	0.7790 (100%) [–]	4.6230 (100%) [–]	42.2062 (64%) [52.8, 75.2]	414.6083 (20%) [10.7, 29.3]	47.1146
5,000	2.7112 (100%) [–]	19.7426 (100%) [–]	157.0027 (46%) [34.4,57.6]	1,767.7945 (16%) [7.5, 24.5]	99.5008

(%xxx) = Percent winning in computing time of the algorithm 1 (or algorithm 2) compared to the Monte Carlo simulation for solving SKPCRCR.

[xx.x, xx.x] = 90% confidence interval of percent winning in computing time

[–] = Cannot find the confidence interval because of percent winning in computing time is 100%

According to Tables 10 and 11, the computing time (sec) of the algorithm 1 (or algorithm 2) was smaller than the computing time of the Monte Carlo simulation, when $\Delta B = 0.01\sum_{j=1}^{n} a_j t_j$ and $0.001\sum_{j=1}^{n} a_j t_j$ for all n and when $\Delta B = 0.0001\sum_{j=1}^{n} a_j t_j$, the computing time (sec) of the algorithm 1 (or algorithm 2) was smaller than the computing time of the Monte Carlo simulation for $n \le 2500$. Furthermore, for $\Delta B = 0.01\sum_{j=1}^{n} a_j t_j$ and $0.001\sum_{j=1}^{n} a_j t_j$, the percent winning in computing time is 100% for or all *n*. And the percent winning decreased when ΔB is more delicate and also decrease when *n* increased.

Again, fifty generated samples were conducted to compare the quality of solutions between using the algorithm 1 (or algorithm 2) and the Monte Carlo simulation for solving the relaxed problem. The percent of Win/Tie/Lose of the objective values of SKPCRCR by using the algorithm 1 (or algorithm 2) of SKPCRCR compared with the Monte Carlo simulation of SKPCRCR were presented in Table 12. The percent of Win/Tie/Lose of the objective values of SKPCRC where the relaxed problem was solved by the algorithm 1 (or algorithm 2) of SKPCRCR compared with the objective values of SKPCRC where the relaxed problem was solved by the Monte Carlo simulation of SKPCRCR were presented in Table 13.

Table 12 Percent of Win/Tie/Lose of the objective values of SKPCRCR by using the algorithm 1 (or algorithm 2) of SKPCRCR compared with the Monte Carlo simulation of SKPCRCR

n	ΔB											
	$0.01\sum_{j=1}^{n} a_j t_j$			$0.001\sum_{j=1}^{n} a_j t_j$			$0.0001\sum_{j=1}^{n} a_j t_j$			$0.00001\sum_{j=1}^{n} a_j t_j$		
	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose
100	76	0	24	94	0	6	100	0	0	100	0	0
250	82	0	18	100	0	0	100	0	0	100	0	0
500	76	0	24	96	0	4	100	0	0	100	0	0
750	80	0	20	96	0	4	100	0	0	100	0	0
1,000	74	0	26	98	0	2	100	0	0	100	0	0
2,500	76	0	24	94	0	6	98	0	2	100	0	0
5,000	68	0	32	98	0	2	100	0	0	100	0	0
Average	76.00	0.00	24.00	96.57	0.00	3.43	99.71	0.00	0.29	100.00	0.00	0.00

According to Table 12, when ΔB was $0.01\sum_{j=1}^{n} a_j t_j$, the percent of win was 76%. And the percent of win was improved to more than 95% when ΔB was more delicate. In addition, when ΔB was $0.00001\sum_{j=1}^{n} a_j t_j$, the percent of win was 100%.

Table 13 Percent of Win/Tie/Lose of the objective values of SKPCRC where the relaxed problem was solved by the algorithm 1 (or algorithm 2) of SKPCRCR compared with the objective values of SKPCRC where the relaxed problem was solved by the Monte Carlo simulation of SKPCRCR.

n	ΔB											
	$0.01\sum_{j=1}^{n} a_j t_j$			$0.001\sum_{j=1}^{n} a_j t_j$			$0.0001\sum_{j=1}^{n} a_j t_j$			$0.00001\sum_{j=1}^{n} a_j t_j$		
	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose
100	34	66	0	32	68	0	32	68	0	32	68	0
250	36	40	24	40	60	0	40	58	2	38	60	2
500	46	28	26	50	48	2	50	50	0	54	46	0
750	60	26	14	70	30	0	70	30	0	70	30	0
1,000	58	12	30	68	28	4	70	30	0	68	32	0
2,500	64	8	28	70	20	10	74	24	2	74	24	2
5,000	60	8	32	82	8	10	84	12	4	84	12	4
Average	51.14	26.86	22.00	58.86	37.43	3.71	60.00	38.86	1.14	60.00	38.86	1.14

According to Table 13, the percent of win were more than 50% for all ΔB. And when ΔB was $0.001\sum_{j=1}^{n} a_j t_j$, $0.0001\sum_{j=1}^{n} a_j t_j$ and $0.00001\sum_{j=1}^{n} a_j t_j$, the percent of not losing (either win or tie) were more than 95%. Furthermore, when *n* increased the percent of win tended to be increased.

The average absolute percent error was calculated in order to represent how far of the objective value of the algorithm 1 (or algorithm 2) compared with the Monte Carlo simulation. Average absolute percent error of the objective values of SKPCRCR by using the algorithm 1 (or algorithm 2) of SKPCRCR compared with the Monte Carlo simulation of SKPCRCR were presented in Table 14. And the average absolute percentage error of the objective values of SKPCRC where the relaxed problem was solved by the algorithm 1 (or algorithm 2) of SKPCRCR compared with the average absolute percent error of the objective values of SKPCRC where the relaxed problem was solved by the Monte Carlo simulation of SKPCRCR were presented in Table 15.

Table 14 Average absolute percent error of the objective values of SKPCRCR by using the algorithm 1 (or algorithm 2) of SKPCRCR compared with the Monte Carlo simulation of SKPCRCR

n	ΔB							
	$0.01\sum_{j=1}^{n} a_j t_j$		$0.001\sum_{j=1}^{n} a_j t_j$		$0.0001\sum_{j=1}^{n} a_j t_j$		$0.00001\sum_{j=1}^{n} a_j t_j$	
	Win	Lose	Win	Lose	Win	Lose	Win	Lose
100	6.2311	8.9118	5.1110	0.0858	4.8230	0.0000	4.8231	0.0000
250	6.1705	6.5055	5.4439	0.0000	5.4539	0.0000	5.4540	0.0000
500	7.6027	5.5935	6.6293	0.0205	6.3686	0.0000	6.3687	0.0000
750	6.9280	5.2874	6.7718	0.0540	6.5150	0.0000	6.5152	0.0000
1,000	10.0321	12.2070	7.8805	0.0854	7.7525	0.0000	7.7529	0.0000
2,500	9.4491	11.3018	8.3002	0.0607	8.2090	0.0001	8.0456	0.0000
5,000	9.7013	7.6606	8.2198	0.1999	8.0845	0.0000	8.0847	0.0000
Average	8.0164	8.2097	6.9081	0.0723	6.7438	0.0000	6.7206	0.0000

According to Table 14, for $\Delta B = 0.01\sum_{j=1}^{n} a_j t_j$, the values of average absolute percent error of the winning cases and losing cases were not much different. However, for $\Delta B = 0.001\sum_{j=1}^{n} a_j t_j$ and $0.0001\sum_{j=1}^{n} a_j t_j$,the average absolute percent error of the winning cases were significantly greater than the average absolute percent error of the losing cases, i.e. for $\Delta B = 0.00001\sum_{j=1}^{n} a_j t_j$, there was no losing case at all. Moreover, for $\Delta B = 0.001\sum_{j=1}^{n} a_j t_j$, $0.0001\sum_{j=1}^{n} a_j t_j$ and $0.00001\sum_{j=1}^{n} a_j t_j$, the values of the average absolute percent error of the winning cases tended to increase when n increased. If scale of ΔB was more delicate, the average absolute percent error of losing cases significantly decreased but the average absolute percent error of the winning cases only slightly decreased.

Table 15 Average absolute percent error of the objective values of SKPCRC where the relaxed problem was solved by the algorithm 1 (or algorithm 2) of SKPCRCR compared with average absolute percent error of the objective values of SKPCRC where the relaxed problem was solved by the Monte Carlo simulation of SKPCRCR

n	ΔB							
	$0.01\sum_{j=1}^{n} a_j t_j$		$0.001\sum_{j=1}^{n} a_j t_j$		$0.0001\sum_{j=1}^{n} a_j t_j$		$0.00001\sum_{j=1}^{n} a_j t_j$	
	Win	Lose	Win	Lose	Win	Lose	Win	Lose
100	5.4844	0.0000	5.8574	0.0000	5.8574	0.0000	5.8574	0.0000
250	5.5458	0.2132	5.1403	0.0000	5.1382	0.0688	5.4087	0.0688
500	4.5674	0.7123	4.3098	0.0098	4.3098	0.0000	4.1440	0.0000
750	3.8206	2.5014	4.1129	0.0000	4.1129	0.0000	4.1129	0.0000
1,000	5.7348	3.7699	4.2893	0.0761	4.8483	0.0000	4.8483	0.0000
2,500	4.8014	1.2767	4.2655	0.0268	4.3284	0.0015	4.3284	0.0015
5,000	4.7165	3.8174	3.9645	0.0059	3.8709	0.0010	3.8709	0.0010
Average	4.9530	1.7559	4.5628	0.0169	4.6380	0.0102	4.6529	0.0102

According to Table 15, when $\Delta B = 0.01\sum_{j=1}^{n} a_j t_j$, the values of the average absolute percent error of the winning cases were significantly greater than the average absolute percent error of the losing cases for small n (i.e., $n \le 500$). However, the values of the average absolute percent error of the winning cases were only slightly greater than the average absolute percent error of the losing case for large n (i.e., $n \ge 750$). When $\Delta B = 0.001\sum_{j=1}^{n} a_j t_j$, $0.0001\sum_{j=1}^{n} a_j t_j$ and $0.00001\sum_{j=1}^{n} a_j t_j$, the values of the average absolute percent error of the winning case were significantly greater than the average absolute percent error of the losing cases for all n. If scale of ΔB was more delicate, the average absolute percent error of the losing cases tended to be decreased.

In order to select the appropriate algorithm, the computing time and the quality of the solutions are important criteria. From Tables 10 – 15, it can be

concluded that the algorithm 1 (or algorithm 2) with appropriate ΔB was superior to the Monte Carlo simulation. The summary of the algorithm 1 (or algorithm 2) of SKCRCR with appropriate ΔB compared with the Monte Carlo simulation for solving SKPCRCR was shown in Table 16. And the summary of the proposed heuristic where the relaxed problem was solved by the using algorithm 1 (or algorithm 2) of SKPCRCR with appropriate ΔB compared with the proposed heuristic where the relaxed problem was solved by using the Monte Carlo simulation.

Table 16 Summary of the algorithm 1 (or algorithm 2) of SKPCRCR with appropriate ΔB compared with the Monte Carlo simulation for solving SKPCRCR

n	Appropriate ΔB	SKPCRCR						
		Percent of Objective Value		Aver. Abs. Percent Err.		Computing Time	Ratio of Computing Time	Percent Win of Computing Time
		Win	Lose	Win	Lose			
100	$0.0001\sum_{j=1}^{n} a_j t_j$	100 [–]	0 [–]	4.823	0.000	1.136	4.051	100 [–]
250	$0.0001\sum_{j=1}^{n} a_j t_j$	100 [–]	0 [–]	5.454	0.000	2.381	2.994	100 [–]
500	$0.0001\sum_{j=1}^{n} a_j t_j$	100 [–]	0 [–]	6.369	0.000	5.126	2.161	96 [91.4, 100.0]
750	$0.0001\sum_{j=1}^{n} a_j t_j$	100 [–]	0 [–]	6.515	0.000	7.584	2.093	88 [80.4, 95.6]
1,000	$0.0001\sum_{j=1}^{n} a_j t_j$	100 [–]	0 [–]	7.752	0.000	11.003	1.764	80 [70.7, 89.3]
2,500	$0.0001\sum_{j=1}^{n} a_j t_j$	98 [94.7, 100.0]	2 [0.0,5.3]	8.209	0.000	41.929	1.118	64 [52.8, 75.2]
5,000	$0.001\sum_{j=1}^{n} a_j t_j$	98 [94.7, 100.0]	2 [0.0,5.3]	8.220	0.200	18.847	5.227	100 [–]

The ratios of computing time was calculated from taking the computing time of the algorithm 1 (or algorithm 2) of SKPCRCR divided by the computing time of the Monte Carlo simulation for solving SKPCRCR.

Table 17 Summary of the proposed heuristic where the relaxed problem was solved by using the algorithm 1 (or algorithm 2) of SKPCRCR with appropriate ΔB compared with the proposed heuristic where the relaxed problem was solved by using the Monte Carlo simulation

n	Appropriate ΔB	SKPCRC						
		Percent of Objective Value		Aver. Abs. Percent Err.		Computing Time	Ratio of Computing Time	Percent Win of Computing Time
		Win	Lose	Win	Lose			
100	$0.0001\sum_{j=1}^{n} a_j t_j$	32 [21.4, 42.5]	0 [–]	5.857	0.000	1.151	4.010	100 [–]
250	$0.0001\sum_{j=1}^{n} a_j t_j$	40 [28.6, 53.4]	2 [0.0,5.3]	5.138	0.069	2.404	2.925	100 [–]
500	$0.0001\sum_{j=1}^{n} a_j t_j$	50 [38.4, 61.6]	0 [–]	4.310	0.000	5.161	2.155	96 [91.4, 100.0]
750	$0.0001\sum_{j=1}^{n} a_j t_j$	70 [59.3, 80.7]	0 [–]	4.113	0.000	7.629	2.032	88 [80.4, 95.6]
1,000	$0.0001\sum_{j=1}^{n} a_j t_j$	70 [59.3, 80.7]	2 [0.0,5.3]	4.848	0.002	11.072	1.758	80 [70.7, 89.3]
2,500	$0.0001\sum_{j=1}^{n} a_j t_j$	74 [63.8, 82.2]	2 [0.0,5.3]	3.965	0.006	42.206	1.116	64 [52.8, 75.2]
5,000	$0.001\sum_{j=1}^{n} a_j t_j$	82 [73.1, 90.9]	10 [3.0,17.0]	8.220	0.200	19.743	5.040	100 [–]

[xx.x, xx.x] = 90% Confidence interval of the percent win/lose

[–] = Cannot find the confidence interval because of percent of win/lose is 0% or 100%

The ratio of computing time was calculated from taking the computing time of the proposed heuristic where the relaxed problem was solved by using the algorithm 1 (or algorithm 2) of SKPCRCR divided by the computing time of the proposed heuristic where the relaxed problem was solved by using the Monte Carlo simulation.

For n = 5,000, if the quality of the solutions is more important criteria than the computing time, then better alternative is $\Delta B = 0.0001\sum_{j=1}^{n} a_j t_j$. The computing time of

this ΔB was about 57.5 seconds more than the computing time of the Monte Carlo simulation for both SKPCRCR and SKPCRC. However, the percent of win of SKPCRCR was increased from 98% to 100% while the percent of lose of SKPCRCR was decreased from 2% to 0%. In addition, the percent of win of SKPCRC was increased from 82% to 84% while the percent of lose of SKPCRC was decreased from 10% to 4%.

CONCLUSION AND RECOMMENDATION

Conclusion

The principle objective of this research was to propose the procedures for solving stochastic knapsack problem with discrete and continuous random capacity. For the proposed procedure, the relaxed integer constraints were employed. Next we solved the relaxed problem. Then the optimal solution of the relaxed problem was found as the initial solution for finding the final integer solution.

For the stochastic knapsack problem with discrete random capacity (SKPDRC), two algorithms were proposed for solving the relaxed problem (SKPDRCR) and compared with the general purpose method using CPLEX. The fifty generated samples were generated to compare the computing time for large problems (i.e., $m \geq 2,500$ and $n \geq 2,500$). Otherwise, a hundred generated samples were generated. As the results, the algorithm 2 of SKPDRCR had shorter computing time than both the algorithm 1 of SKPDRCR and the general purpose method using CPLEX. Therefore, we used the algorithm 2 of SKPDRCR to find the initial solution for the proposed heuristic. The proposed heuristic for SKPDRC where the relaxed problem was solved by using the algorithm 2 of SKPDRCR was compared with the general purpose method using CPLEX. The results were shown that the proposed heuristic for SKPDRC had shorter computing time than the general purpose method using CPLEX. The fifty generated samples were generated to compare the quality of integer solutions (i.e., comparing objective values) between the proposed heuristic for SKPDRC and the general purpose method using CPLEX with the default settings (i.e., MIP gap is 0.01%). The percent of win of the proposed heuristic for SKPDRC was more than the percent of lose and percent of win was significantly more than percent of lose when n was large. However, the percent of lose of the proposed heuristic for SKPDRC was 1.0667% where 90% confidence interval was [1.0496%, 2.2284%].

For the stochastic knapsack problem with continuous random capacity (SKPCRC), two algorithms were proposed for solving the relaxed problem (SKPCRCR) and compared with the Monte Carlo simulation. According to the proof of the algorithms 1 and 2 of SKPCRCR, the criteria of these two algorithms were the same. The fifty generated samples were generated to compare an efficiency (i.e., computing time) and effectiveness (i.e., quality of solutions) between the algorithm 1 (or algorithm 2) and the Monte Carlo simulation.

In order to select the appropriate algorithm for SKPCRCR and SKPCRC, the computing time and the quality of the solutions were important criteria.

For the algorithm 1 (or algorithm 2) of SKPCRCR, if ΔB was more delicate, the computing time increased but it gave the better solution. As the results, the algorithm 1 (or algorithm 2) of SKPCRCR with appropriate ΔB was superior to the Monte Carlo simulation of SKPCRCR. For $n \le 2,500,$ the appropriate ΔB was $0.0001\sum_{j=1}^{n} a_j t_j$. For $n \ge 5,000,$ the appropriate ΔB was $0.001\sum_{j=1}^{n} a_j t_j$.

For SKPCRC where the relaxed problem was solved by using the algorithm 1 (or algorithm 2) of SKPCRCR, if ΔB was more delicate, the computing time increased but it gave better solution. As the results, the proposed heuristic for solving SKPCRC where the relaxed problem was solved by using the algorithm 1 (or algorithm 2) of SKPCRCR with appropriate ΔB was superior to the proposed heuristic for solving SKPCRC where the relaxed problem was solved by using the Monte Carlo simulation. For $n \le 2,500,$ the appropriate ΔB was $0.0001\sum_{j=1}^{n} a_j t_j$. For $n \ge 5,000,$ the appropriate ΔB was $0.001\sum_{j=1}^{n} a_j t_j$.

Recommendation

In this study, there was 1.0667% that the proposed heuristic for solving SKPDRC lost to CPLEX. Moreover, there were few cases that the algorithm 1 (or algorithm 2) of SKPCRC lost to the Monte Carlo simulation. Therefore, it is worth to study the characteristic types of generated data, which make the integer solution of the proposed heuristic for SKPDRC worse than the integer solution of CPLEX. In addition, the study the characteristic types of generated data, which make the integer solution of the proposed heuristic for SKPCRC where the relaxed solution was solved by using the algorithm 1 (or algorithm 2) of SKPCRCR worse than integer solution of the proposed heuristic for SKPCRC where the relaxed solution was solved by using the Monte Carlo simulation also should be investigated.

Furthermore, it is worth to study the other types of stochastic knapsack problem such as uncertain in both benefit and capacity, and uncertain in both weight and capacity.

LITERATURE CITED

Andonov, R., V. Poirriez and S. Rajopadhye. 2000. Unbounded Knapsack Problem: Dynamic Programming Revisited. **European Journal of Operational Research.** 123: 394-407.

Andradottir, S. 1996. A Global Search Method for Discrete Stochastic Optimization. **SIAM Journal on Optimization.** 513-530.

Audsley, E., S. Barker, K. Darby-Dowman and D. Parsons. 2000. A Two-Stage Stochastic Programming with Recourse Model for Determining Robust Planting Plans in Horticulture. **The Journal of the Operational Research Society.** 51: 83-89.

Balas, E., F. Glover and S. Zionts. 1965. An Additive Algorithm for Solving Linear Programs with Zero-One Variables. **Operations Research.** 13: 517-546.

Balas, E. and E. Zemel. 1980. An Algorithm for Large Zero-One Knapsack Problems. **Operations Research.** 28: 1130-1154.

Beasly, J.E. and P.C. Chu. 1998. A Genetic Algorithm for the Multidimensional Knapsack Problem. **Journal of Heuristic.** 4: 63-86.

Bellman, R.E. 1957. **Dynamic Programming.** Princeton University Press, Princeton, New Jersey.

Birge, J.R. and F. Louveaux. 1997. **Introduction to Stochastic Programming.** Springer, New York.

Chiu, L.A., Jr. Cox and L.L. Lu. 1999. Optimal Project Selection: Stochastic Knapsack with Finite Time Horizon. **The Journal of the Operational Research Society.** 50: 645-650.

Dantzig, G.B. 1955. Linear Programming under Uncertainty. **Management Science.** 1: 197-206.

_____. and P. Glynn. 1990. Parallel Processors for Planning Under Uncertainty. **Annals of Operations Research.** 22: 1-21.

Drexl, A. 1988. A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem. **Computing.** 40: 1-8.

Dyer, H.E. 1980. Calculating surrogate constraints. **Mathematical Programming.** 19: 255-278.

Edmundson, H.P. 1956. Bounds on the Expectation of a Convex Function of a Random Variable. **RAND Corporation Paper 982**, Santamonica, California.

Elkihel, M. and G. Plateau. 1985. A Hybrid Algorithm for the 0-1 Knapsack Problem. **Methods of Operations Research.** 49: 277-293.

Elmaghraby, S.E. 1959. An Approach to Linear Programming under Certainty. **Operations Research.** 7: 208-216.

_____. 1960. Allocation under Uncertainty When the Demand has Continuous D.F. **Management Science.** 6: 270-294.

Ermoliev, Y. M. 1983. Stochastic Quasigradient Methods and Their Applications to Systems Optimization. **Stochastics.** 9: 1-36.

_____, V.I. Norkin and A. Ruszczynski. 1998. On Optimal Allocation of Invisibles under Uncertainty. **Operations Research.** 46: 381-395.

Everett, H. 1963. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. **Operations Research.** 11: 399-417.

Fayard, D. and Plateau, G. 1975. Resolution of the 0-1 Knapsack Problem: Comparison of Methods. **Mathematical Programming.** 8:272-307.

_____ and _____. 1982. Algorithm for the Solution of the 0-1 Knapsack Problem. **Computing.** 28: 269-287.

Frank, M. and P. Wolfe. 1956. An Algorithm for Quadratic Programming. **Naval Research Logistics Quarterly.** 3: 95-110.

Freville, A. and S. Hanafi. 1998. An Efficient Tabu Search Approach for the 0-1 Multidimensional Knapsack Problem. **European Journal of Operational Research.** 106: 659-675.

_____ and G. Plateau. 1993. An Exact Search for the Solution of the Surrogate Dual of the 0-1 Bidimensional Knapsack Problem. **European Journal of Operational Research.** 68: 413-421.

Garfinkel, R. and G. Nemhauser. 1972. Integer Programming. Wiley, New York.

Garstka, S.J. and D.P. Rutenberg. 1970. Computation in Discrete Stochastic Programs with Recourse. **Operations Research.** 21: 112-122.

Gavish, B. and H. Pirkul. 1985. Efficient Algorithms for Solving Multiconstraint Zero-One Knapsack Problems to Optimality. **Mathematical Programming.** 31: 78-105.

Gilmore P.C. and R.E. Gomory. 1966. The Theory and Computation of Knapsack Functions. **Operations Research**. 14: 1045-1075.

Glover, F. 1965. A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem. **Operations Research.** 13:879-919.

_____. 1968. Surrogate Constraints. Operations Research. 16: 741-749.

_____. 1975. Surrogate Constraints Duality in Mathematical Programming. Operations Research. 23: 434-451.

_____ and G.A. Kochenberger. 1996. Critical Event Tabu Search for Multidimensional Knapsack Problems, *In* I.H. Osman and J.P. Kelly, ed. **Metaheuristics: The Theory and Applications**, Kluwer Academic Publishers. 407-427.

Greenberg, H. and R. Hegerich. 1970. A Branch Search Algorithm for the Knapsack Problem. **Management Science.** 16: 327-332.

Guang, M.D. , L.L. Ken and H.L. Qing. 2003. A Genetic Algorithm for the Unbounded Knapsack Problem. **Proceedings of the Second International Conference on Machine Learning and Cybernatics.** 3: 1586-1590.

Han, K.H., C.H. Lee and K.H. Park. 2001. Parallel Quantum-Inspired Genetic Algorithm for Combinatorial Optimization Problem. **Proceedings of the 2001 IEEE Congress on Evolutionary Computation-Seoul**, Korea. 27-30.

Henig, M.I. 1990. Risk Criteria in a Stochastic Knapsack Problem. **Operations Research**. 38: 820-825.

Higle, J. and S. Sen. 1991. Stochastic Decomposition: An Algorithm for Two Stage Linear Programs with Recourse. **Mathematics of Operations Research.** 16: 650-669.

Horowitz, E. and Sahni, S. 1974. Computing Partitions with Applications to the Knapsack Problem. **Journal of ACM.** 21: 277-292.

Hu, T.C. 1969. **Integer Programming and Network Flows.** Addison-Wesley, Reading, Massachusetts.

ILOG 2002. **CPLEX 8.1.** (Computer Program). (Available at http://www.ilog.com/products/cplex).

Ingargiola, G.P. and J.F. Korsh. 1973. Reduction Algorithm for Zero-One Single Knapsack Problems. **Management Science.** 20: 460-463.

_____ and _____. 1977. A General Algorithm for One-Dimensional Knapsack Problems. **Operations Research.** 25: 752-759.

Kall, P. and S.W. Wallace. 1994. **Stochastic Programming.** John Wiley, New York.

Kaplan, S. 1966. Solution of the Lories-Savage and Similar Integer Programming Problem by the Generalized Lagrange Multiplier Method. **Operations Research.** 14: 1130-1136.

Karwan, M.H. and R.L. Radin. 1984. Surrogate Dual Multiplier Search Procedures in Integer Programming. **Operations Research.** 32: 52-69.

Kleywegt, A.J. and J.D. Papastavrou. 1998. The Dynamic and Stochastic Knapsack Problem. **Operations Research.** 46: 17-35.

_____ and_____. 2001. The Dynamic and Stochastic Knapsack Problem with Random Sized Items. **Operations Research.** 49: 26-41.

_____, _____ and S. Rajagopalan. 1996. The Dynamic and Stochastic Knapsack Problem with Deadlines. **Management Science.** 42: 1706-1718.

Kolesar, P.J. 1967. A Branch and Bound Algorithm for the Knapsack Problem. **Management Science.** 13: 723-735.

Lories, J. and L.J. Savage. 1955. Three Problems in Capital Rationing. **Journal of Business.** 28: 229-239.

Madansky, A. 1959. Bounds on the Expectation of a Convex Function of a Multivariate Random Variable. **Annals of Mathematical Statistics.** 30: 743-746.

Marsman, S. and A. Volgenant. 1998. A Core Approach to the 0-1 Equality Knapsack Problem**. Operational Research Society.** 49: 86-92.

Marsten, R.E. and T.L. Morin. 1978. A Hybrid Approach to Discrete Mathematical Programming. **Mathematical Programming.** 14: 21-40.

Martello, S. and P. Toth. 1977a. An Upper Bound for the 0-1 Knapsack Problem and a Branch and Bound Algorithm. European Journal of Operational Research. 1: 169-175.

_____ and _____. 1977b. Branch and Bound algorithm for the Solution of the General Unidimensional Knapsack Problem. *In* M. Roubens, ed. **Advances in Operations Research.** North-Holland, Amsterdam.

_____ and _____. 1981. A Branch and Bound algorithm for Zero-One Multiple Knapsack Problem. **Discrete Applied Mathematics.** 3: 275-288.

_____ and _____. 1984. A Mixture of Dynamic Programming and Branch-and-Bound for the Subset-Sum Problem. Management Science. 30: 765-771.

_____ and _____. 1988. A New Algorithm for the 0-1 Knapsack Problem. **Management Science.** 34: 633-644.

_____ and _____. 1990. **Knapsack Problems: Algorithm and Computer Implementations.** John Wiley & Sons, West Sussex, England.

_____ and _____. 1997. Upper Bounds and Algorithms for Hard 0-1 Knapsack Problems. **Operations Research.** 45: 768-778.

_____ and _____. 1999. Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem. **Management Science.** 45: 414-424.

Martello, S., D. Pisinger and P. Toth. 1999. Dynamic Programming and Strong Bound for 0-1 Knapsack Problem. **Management Science.** 45: 414-424.

_____, _____ and _____. 2000. New Trends in Exact Algorithms for the 0-1 Knapsack Problem. **European Journal of Operational Research.** 123: 325-332.

Mukai, H. and D. Yan. 1992. Stochastic Discrete Optimization. **SIAM Journal on Control and Optimization.** 30: 594-612.

Nauss, R.M. 1976. An Efficient Algorithm for the 0-1 Knapsack Problem. **Management Science.** 23: 27-31.

Nemhauser, G.L. and L.A. Wolsey. 1988. **Integer and Combinatorial Optimization.** Wiley, New York.

_____ and Z. Ullmann. 1969. Discrete Dynamic Programming and Capital Allocation. **Management Science.** 15: 494-505.

Pandit, S.N.N. and K. Ravi. 1993. A Lexicographic Search for Strongly Correlated 0-1 Knapsack Problems**. Opsearch.** 30: 97-116.

Parks, M.S. and E. Steinberg. 1979. A Preference Order Dynamic Program for a Knapsack Problem with Stochastic Rewards. **Journal of the Operation Research Society.** 30: 141-147.

Pferschy, U. 1999. Dynamic Programming Revisited: Improving Knapsack Algorithms.T **Computing.** 63: 419-430.

Pisinger, D. 1995. An Expanding Core Algorithm for the Exact Knapsack Problem. **Eurpoean Journal of Operational Research.** 87:175-187.

_____. 1997. A Minimal Algorithm for the 0-1 Knapsack Problem, **Operations Research.** 45: 758-767.

_____. 1999a. An Exact Algorithm for Large Multiple Knapsack Problems. **European Journal of Operational Research.** 114: 528-541.

_____. 2000. A Minimal Algorithm for the Bounded Knapsack Problem. **INFORM Journal on Computing.** 12: 75-82.

Ram, B. and S. Sarin. 1988. An Algorithm for the 0-1 Equality Knapsack Problem. **The Journal of the Operation Research Society.** 39: 1045-1049.

Shih, W. 1979. A Branch and Bound Method for the Multiconstraint Zero-One Knapsack Problem. **The Journal of the Operation Research Society.** 30: 369- 378.

Slyke, R.V. and Y. Young. 2000. Finite Horizon Stochastic Knapsacks with Applications to Yield Management. **Operations Research.** 48: 155-172.

Sniedovich, M. 1980. Preference Order Stochastic Knapsack Problems: Methodological Issues. **The Journal of the Operation Research Society.** 31: 1025-1032.

Toth, P. 1980. Dynamic Programming Algorithms for the Zero-One Knapsack Problem. **Computing.** 25: 29-45.

Zangwill, W.I. 1967. The Convex Simplex Method. **Management Science.** 14: 221-238.

Ziemba, W.T. 1970. Computational Algorithm for Convex Stochastic Programs with Simple Recourse. **Operations Research.** 18: 414-431.

APPENDICES

Appendix A

Example of solving SKPDRC

Example of solving SKPDRC

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} (gp_i u_i + hp_i v_i)$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u_i - v_i = b_i$$

$$0 \le x_j \le t_j$$

$$\sum_{i=1}^{m} p_i = 1$$

where

$g = 2.8920$

$h = 5.6602$

c = [2.4106 2.2333 3.9110 3.9655 1.3514
1.3479 5.1126 9.2752 1.3257 3.1003]

a = [9.9130 8.4452 8.7136 8.7915 1.8699
1.5953 4.7963 7.1203 0.9290 0.2170]

t = [1.2328 2.6247 3.6969 1.8626 5.1709
1.6094 6.9864 1.1345 4.9600 2.8753]

b = [15.4291 22.9465 52.6662 53.0728 63.9538
79.1871 94.3548 105.4054 147.7476 155.4211]

p = [0.0667 0.0458 0.1753 0.0675 0.0869
0.1385 0.1441 0.0621 0.1436 0.0695]

Algorithm 1 for Solving SKPDRCR

Calculate c/a and $\sum a_j t_j$.

c/a = [0.2432 0.2644 0.4488 0.4511 0.7227
0.8449 1.0659 1.3026 1.4270 14.2871]

$\sum a_j t_j$ = [12.2207 34.3869 66.6002 82.9752 92.6443
95.2118 128.7206 136.7986 141.4065 142.0304]

Check for case I.

$c_1 / a_1 = 0.2432 < g$ $\therefore$ not case I

Check for case II.

$c_n / a_n = 14.2871$

$\sum_{j=1}^{n} a_j t_j = 142.0304$

$\therefore i = 1, 2, ..., 8 \in B_1$

$i = 9, 10 \in B_2$

$$g\sum_{i\in B_1} p_i - h\sum_{i\in B_2} p_i = 2.8920(0.0667 + 0.0458 + 0.1753 + 0.0675 + 0.0869 + 0.1385 + 0.1441+0.0621)-5.6602(0.1436+0.0695)$$

$$= -3.8377 < c_n / a_n \quad \therefore \text{not case II}$$

Check for case III.

$q = m / 2 = 5$

$b_5 = 63.9538$

$\sum_{j=1}^{2} a_j t_j = 34.3869 < b_5$

$\sum_{j=1}^{3} a_j t_j = 66.6002 > b_5$

$\therefore k = 3$

$$cgh1 = (-c_k / a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i)$$

$$= -0.4488 + 2.8920(0.0869 + 0.1385 + 0.1441 + 0.0621 + 0.1436 + 0.0695) - 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675)$$

$$= -0.5954 < 0$$

$$cgh2 = (c_k / a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$$

$$= 0.4488 + 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675 + 0.0869) - 2.8920(0.1385 + 0.1441 + 0.0621 + 0.1436 + 0.0695)$$

$$= 1.3386 > 0$$

$q = 6$

$b_6 = 79.1871$

$\sum_{j=1}^{3} a_j t_j = 66.6002 < b_6$

$\sum_{j=1}^{4} a_j t_j = 82.9752 > b_6$

$\therefore k = 4$

$$cgh1 = (-c_k / a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i)$$

$$= -0.4511 + 2.8920(0.1385 + 0.1441 + 0.0621 + 0.1436 + 0.0695)$$
$$- 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675 + 0.0869)$$
$$= -1.3408 < 0$$

$$cgh2 = (c_k / a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$$

$$= 0.4511 + 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675 + 0.0869 + 0.1385)$$
$$- 2.8920(0.1441 + 0.0621 + 0.1436 + 0.0695)$$
$$= 2.5253 > 0$$

$q = 4$

$b_4 = 53.0728$

$\sum_{j=1}^{2} a_j t_j = 34.3869 < b_4$

$\sum_{j=1}^{3} a_j t_j = 66.6002 > b_4$

$\therefore k = 3$

$$cgh1 = (-c_k / a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i)$$

$$= -0.4488 + 2.8920(0.0675 + 0.0869 + 0.1385 + 0.1441 + 0.0621 + 0.1436 +$$
$$0.0695) - 5.6602(0.0667 + 0.0458 + 0.1753)$$
$$= -0.0182 < 0$$

$$cgh2 = (c_k / a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$$

$$= 0.4488 + 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675)$$
$$- 2.8920(0.0869 + 0.1385 + 0.1441 + 0.0621 + 0.1436 + 0.0695)$$
$$= 0.5954 > 0$$

$q = 7$

$b_7 = 94.3548$

$\sum_{j=1}^{5} a_j t_j = 92.6443 < b_7$

$\sum_{j=1}^{6} a_j t_j = 95.2118 > b_7$

$\therefore k = 6$

$$cgh1 = (-c_k / a_k + g\sum_{i=q}^{m} p_i - h\sum_{i=1}^{q-1} p_i)$$

$$= - 0.8449 + 2.8920(0.1441 + 0.0621 + 0.1436 + 0.0695)$$
$$- 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675 + 0.0869 + 0.1385)$$
$$= 1.4810 > 0$$

$$cgh2 = (c_k / a_k + h\sum_{i=1}^{q} p_i - g\sum_{i=q+1}^{m} p_i)$$

$$= 0.8449 + 5.6602(0.0667 + 0.0458 + 0.1753 + 0.0675 + 0.0869 + 0.1385$$
$$+ 0.1441) - 2.8920(0.0621 + 0.1436 + 0.0695)$$
$$= 0.0182 > 0$$

$\because cgh1 > 0$ and $cgh2 > 0$

$\therefore k = 3,\ q = 3,\ b_q = b_3 = 52.6662$

$x_j = t_j,\ \ j = 1, ..., k-1$

$\therefore x_1 = 1.2328$

$x_2 = 2.6247$

$$x_k = \frac{b_q - \sum_{j=1}^{k-1} a_j t_j}{a_k}$$

$$\therefore x_3 = \frac{52.6662 - 34.3869}{8.7136}$$
$$= 2.0978$$

$$x_j = 0, \quad j = k+1, ..., n$$

$$\therefore x_4, x_5, x_6, x_7, x_8, x_9, x_{10} = 0$$

$$u_i = 0 \text{ and } v_i = b_q - b_i, \; i = 1, 2, ..., q-1$$

$$\therefore u_1, u_2 = 0$$
$$v_1 = b_3 - b_1 = 52.6662 - 15.4291 = 37.2371$$
$$v_2 = b_3 - b_2 = 52.6662 - 22.9465 = 29.7197$$

$$u_q = 0 \text{ and } v_q = 0$$

$$\therefore u_3 = 0 \text{ and } v_3 = 0$$

$$u_i = b_i - b_q \text{ and } v_i = 0, \; i = q+1, q+2, ..., m$$

$$\therefore u_4 = b_4 - b_3 = 53.0728 - 52.6662 = 0.4066$$
$$u_5 = b_5 - b_3 = 63.9538 - 52.6662 = 11.2876$$
$$u_6 = b_6 - b_3 = 79.1871 - 52.6662 = 26.5209$$
$$u_7 = b_7 - b_3 = 94.3548 - 52.6662 = 41.6886$$
$$u_8 = b_8 - b_3 = 105.4054 - 52.6662 = 52.7392$$
$$u_9 = b_9 - b_3 = 147.7476 - 52.6662 = 95.0814$$
$$u_{10} = b_{10} - b_3 = 155.4211 - 52.6662 = 102.7549$$
$$v_4, v_5, v_6, v_7, v_8, v_9, v_{10} = 0$$

$$f = \sum_{j=1}^{k-1} c_j t_j + c_k((b_q - \sum_{j=1}^{k-1} a_j t_j)/a_k) + h\sum_{i=1}^{q-1} p_i(b_q - b_i) + g\sum_{i=q+1}^{m} p_i(b_i - b_q)$$

$$= \sum_{j=1}^{k-1} c_j t_j + c_k x_k + h\sum_{i=1}^{q-1} p_i v_i + g\sum_{i=q+1}^{m} p_i u_i$$

$$= 2.4106(1.2328) + 2.2333(2.6247) + 3.9110(2.0978)$$
$$+ 5.6602[0.0667(37.2371) + 0.0458(29.7197)]$$
$$+ 2.8920[0.0675(0.4066) + 0.0869(11.2876) + 0.1385(26.5209)$$
$$+ 0.1441(41.6886) + 0.0621(52.7392) + 0.1436(95.0814) + 0.0695(102.7549)]$$

$= 139.3240$

Optimal solution for SKPDRCR is as follows.

$x^*_{SKPDRCR} = [1.2328 \quad 2.6247 \quad 2.0978 \quad 0.0000 \quad 0.0000$
$0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000]$

$u^*_{SKPDRCR} = [0.0000 \quad 0.0000 \quad 0.0000 \quad 0.4066 \quad 11.2876$
$26.5209 \quad 41.6886 \quad 52.7392 \quad 95.0814 \quad 102.7549]$

$v^*_{SKPDRCR} = [37.2371 \quad 29.7197 \quad 0.0000 \quad 0.0000 \quad 0.0000$
$0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000]$

$k^*_{SKPDRCR} = 3$

$b^*_{SKPDRCR} = 52.6602$

$f^*_{SKPDRCR} = 139.3240$

Algorithm 2 for Solving SKPDRCR

Calculate c' and t'.

$c' = c / a$

$\therefore c' = [0.2432 \quad 0.2644 \quad 0.4488 \quad 0.4511 \quad 0.7227$
$0.8449 \quad 1.0659 \quad 1.3026 \quad 1.4270 \quad 14.2871]$

$t' = a * t$

$\therefore t' = [12.2207 \quad 22.2207 \quad 32.2133 \quad 16.3750 \quad 9.6691$
$2.5675 \quad 33.5089 \quad 8.0780 \quad 4.6078 \quad 0.6239]$

$sumtp_d = \sum_{j=1}^{d} t'_j \quad , d = 1,...,n$

$\therefore sumtp = [12.2207 \quad 34.3869 \quad 66.6002 \quad 82.9752 \quad 92.6443$
$95.2118 \quad 128.7206 \quad 136.7986 \quad 141.4065 \quad 142.0304]$

$ep_d = \sum_{j=1}^{d} t'_j \qquad , d = 1,...,n$

$ep_d = b_i \qquad , (d,i) = (n+1,1),...,(n+m,m)$

$ep_d = 0 \qquad , d = n+m+1$

$\therefore ep$ = [12.2207 34.3869 66.6002 82.9752 92.6443
95.2118 128.7206 136.7986 141.4065 142.0304
15.4291 22.9465 52.6662 53.0728 63.9538
79.1871 94.3548 105.4054 147.7476 155.4211
0]

$eps = \text{sort}(ep)$

$\therefore eps$ = [0 12.2207 15.4291 22.9465 34.3869
52.6662 53.0728 63.9538 66.6002 79.1871
82.7952 92.6443 94.3548 95.2118 105.4054
128.7206 136.7986 141.4065 142.0304 147.7476
155.4211]

Check each extreme point until find that $SH_{(eps_{r-1}<\theta<eps_r)} > 0$.

$r = 2$

$eps_2 = 12.2207 = sumtp_1$

$\therefore kk = 1$

First element from eps_{r+1} to eps_{n+m+1} that is also in b is $eps_3 = 15.4291 = b_1$.

$\therefore iii = 1$

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kk} - g + (g+h)\sum_{i=1}^{iii-1} p_i$$

$$\therefore SH_{(eps_1<\theta<eps_2)} = c'_1 - g + (g+h)\sum_{i=1}^{1-1} p_i$$

$$= 0.2432 - 2.8920 + (2.9820 + 5.6602)(0)$$

$$= -2.6488 < 0$$

$r = 3$

$eps_3 = 15.4291 = b_1$

$\therefore ii = 1$

First element from eps_{r+1} to eps_{n+m+1} that is also in $sumtp$ is $eps_5 = 34.3869 = sumtp_2$.

$\therefore kkk = 2$

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kkk} - g + (g+h)\sum_{i=1}^{ii-1} p_i$$

$$\therefore SH_{(eps_2<\theta<eps_3)} = c'_2 - g + (g+h)\sum_{i=1}^{1-1} p_i$$

$$= 0.2644 - 2.8920 + (2.9820 + 5.6602)(0)$$

$$= -2.6276 < 0$$

$r = 4$

$eps_4 = 22.9465 = b_2$

$\therefore ii = 2$

First element from eps_{r+1} to eps_{n+m+1} that is also in $sumtp$ is $eps_5 = 34.3869 = sumtp_2$.

$\therefore kkk = 2$

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kkk} - g + (g+h)\sum_{i=1}^{ii-1} p_i$$

$$\therefore SH_{(eps_3<\theta<eps_4)} = c'_2 - g + (g+h)\sum_{i=1}^{2-1} p_i$$

$$= 0.2644 - 2.8920 + (2.9820 + 5.6602)(0.0667)$$

$$= -2.0571 < 0$$

$r = 5$

$eps_5 = 34.3869 = sumtp_2$

$\therefore kk = 2$

First element from eps_{r+1} to eps_{n+m+1} that is also in b is $eps_6 = 52.6662 = b_3$.

$\therefore iii = 3$

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kk} - g + (g+h)\sum_{i=1}^{iii-1} p_i$$

$$\therefore SH_{(eps_4<\theta<eps_5)} = c'_2 - g + (g+h)\sum_{i=1}^{3-1} p_i$$

$$= 0.2644 - 2.8920 + (2.9820 + 5.6602)(0.0067 + 0.0458)$$

$$= -1.6654 < 0$$

$r = 6$

$eps_6 = 52.6662 = b_3$

$\therefore ii = 3$

First element from eps_{r+1} to eps_{n+m+1} that is also in $sumtp$ is $eps_9 = 66.6002 = sumtp_3$.

$\therefore kkk = 3$

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kkk} - g + (g+h)\sum_{i=1}^{ii-1} p_i$$

$$\therefore SH_{(eps_5<\theta<eps_6)} = c'_3 - g + (g+h)\sum_{i=1}^{3-1} p_i$$

$$= 0.4488 - 2.8920 + (2.9820 + 5.6602)(0.0667 + 0.0458)$$

$$= -1.4810 < 0$$

$r = 7$

$eps_7 = 53.0728 = b_4$

$\therefore ii = 4$

First element from eps_{r+1} to eps_{n+m+1} that is also in $sumtp$ is $eps_9 = 66.6002 = sumtp_3$.

$\therefore kkk = 3$

$$SH_{(eps_{r-1}<\theta<eps_r)} = c'_{kkk} - g + (g+h)\sum_{i=1}^{ii-1} p_i$$

$$\therefore SH_{(eps_6<\theta<eps_7)} = c'_3 - g + (g+h)\sum_{i=1}^{4-1} p_i$$

$$= 0.4488 - 2.8920 + (2.9820 + 5.6602)(0.0667 + 0.0458 + 0.1753)$$

$$= 0.0182 > 0$$

$\therefore \theta^* = eps_{r-1} = eps_6 = 52.6662$

$k^* = k$ such that $sumtp_{k-1} \le \theta^*$ and $sumtp_k > \theta^*$

$sumtp_2 = 34.3869 < \theta^*$

$sumtp_3 = 66.6002 > \theta^*$

$\therefore k^* = 3$

$x_j = t_j, \quad j = 1, \ldots, k^* - 1$

$\therefore x_1 = 1.2328$

$x_2 = 2.6247$

$$x_{k^*} = \frac{\theta^* - sumtp_{k^*-1}}{a_{k^*}}$$

$$\therefore x_3 = \frac{52.6662 - 34.3869}{8.7136}$$

$$= 2.0978$$

$x_j = 0, \quad j = k^* + 1, ..., n$

$\therefore x_4, x_5, x_6, x_7, x_8, x_9, x_{10} = 0$

$u_i = 0$ and $v_i = \theta^* - b_i$ for i such that $b_i < \theta^*$

$\therefore u_1, u_2 = 0$

$v_1 = b_3 - b_1 = 52.6662 - 15.4291 = 37.2371$

$v_2 = b_3 - b_2 = 52.6662 - 22.9465 = 29.7197$

$u_i = b_i - \theta^*$ and $v_i = 0$ for i such that $b_i \geq \theta^*$

$\therefore u_3 = 52.6662 - 52.6662 = 0$

$u_4 = b_4 - \theta^* = 53.0728 - 52.6662 = 0.4066$

$u_5 = b_5 - \theta^* = 63.9538 - 52.6662 = 11.2876$

$u_6 = b_6 - \theta^* = 79.1871 - 52.6662 = 26.5209$

$u_7 = b_7 - \theta^* = 94.3548 - 52.6662 = 41.6886$

$u_8 = b_8 - \theta^* = 105.4054 - 52.6662 = 52.7392$

$u_9 = b_9 - \theta^* = 147.7476 - 52.6662 = 95.0814$

$u_{10} = b_{10} - \theta^* = 155.4211 - 52.6662 = 102.7549$

$v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10} = 0$

$$f = \sum_{j=1}^{n} c_j x_j + g \sum_{i=1}^{m} p_i u_i + h \sum_{i=1}^{m} p_i v_i$$

$$= 2.4106(1.2328) + 2.2333(2.6247) + 3.9110(2.0978)$$
$$+ 5.6602[0.0667(37.2371) + 0.0458(29.7197)]$$
$$+ 2.8920[0.0675(0.4066) + 0.0869(11.2876) + 0.1385(26.5209)$$
$$+ 0.1441(41.6886) + 0.0621(52.7392) + 0.1436(95.0814) + 0.0695(102.7549)]$$
$$= 139.3240$$

Optimal solution for SKPDRCR is as follows.

$x^*_{SKPDRCR}$ = [1.2328 2.6247 2.0978 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000]

$u^*_{SKPDRCR}$ = [0.0000 0.0000 0.0000 0.4066 11.2876
26.5209 41.6886 52.7392 95.0814 102.7549]

$v^*_{SKPDRCR}$ = [37.2371 29.7197 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000]

$k^*_{SKPDRCR} = 3$

$b^*_{SKPDRCR} = 52.6602$

$f^*_{SKPDRCR} = 139.3240$

Algorithm 1 and 2 of SKPDRCR has the same optimal solution.

Proposed Heuristic for SKPDRC

$f_{\min} = \infty$

$b^*_{SKPDRCR} = 52.6602$

$k = k^*_{SKPDRCR} = 3$

$tint = \lfloor t \rfloor$ = [1 2 3 1 5 1 6 1 4 2]

$x = \lfloor x^*_{SKPDRCR} \rfloor$ = [1 2 2 0 0 0 0 0 0 0]

$\sum_{j=1}^{n} a_j x_j = 44.2306$

Calculate u_i and v_i.

$u_i = 0$ and $v_i = \sum_{j=1}^{n} a_j x_j - b_i$, for i such that $b_i < \sum_{j=1}^{n} a_j x_j$

$\therefore u_1, u_2 = 0$

$v_1 = \sum_{j=1}^{n} a_j x_j - b_1 = 44.2306 - 15.4291 = 28.8015$

$v_2 = \sum_{j=1}^{n} a_j x_j - b_2 = 44.2306 - 22.9465 = 21.2841$

$u_i = b_i - \sum_{j=1}^{n} a_j x_j$ and $v_i = 0$, for i such that $b_i \geq \sum_{j=1}^{n} a_j x_j$

$\therefore u_3 = b_3 - \sum_{j=1}^{n} a_j x_j = 52.6662 - 44.2306 = 8.4356$

$u_4 = b_4 - \sum_{j=1}^{n} a_j x_j = 53.0728 - 44.2306 = 8.8422$

$u_5 = b_5 - \sum_{j=1}^{n} a_j x_j = 63.9538 - 44.2306 = 19.7232$

$u_6 = b_6 - \sum_{j=1}^{n} a_j x_j = 79.1871 - 44.2306 = 34.9565$

$u_7 = b_7 - \sum_{j=1}^{n} a_j x_j = 94.3548 - 44.2306 = 50.1242$

$u_8 = b_8 - \sum_{j=1}^{n} a_j x_j = 105.4054 - 44.2306 = 61.1748$

$u_9 = b_9 - \sum_{j=1}^{n} a_j x_j = 147.7476 - 44.2306 = 103.5170$

$u_{10} = b_{10} - \sum_{j=1}^{n} a_j x_j = 155.4211 - 44.2306 = 111.1905$

$v_4, v_5, v_6, v_7, v_8, v_9, v_{10} = 0$

$$f = \sum_{j=1}^{n} c_j x_j + g \sum_{i=1}^{m} p_i u_i + h \sum_{i=1}^{m} p_i v_i$$

$= [2.4106(1) + 2.2333(2) + 3.9110(2) + 3.9655(0) + 1.3514(0)$
$1.3479(0) + 5.1126(0) + 9.2752(0) + 1.3257(0) + 3.1003(0)]$
$+ 2.8920[0.0667(0) + 0.0458(0) + 0.1753(8.4356) + 0.0675(8.8422)$
$+ 0.0869(19.7232) + 0.1385(34.9565) + 0.1441(50.1242) + 0.0621(61.1748)$
$+ 0.1436(103.5170) + 0.0695(111.1905)]$
$+ 5.6602[0.0667(28.8015) + 0.0458(21.2841) + 0.1753(0) + 0.0675(0)$
$+ 0.0869(0) + 0.1385(0) + 0.1441(0) + 0.0621(0) + 0.1436(0) + 0.0695(0)]$

$= 153.2649 < f_{\min}$

$\therefore f_{\min} = 153.2649$

$\therefore xint$ [1 2 2 0 0 0 0 0 0 0]

$uint = [0.0000 \quad 0.0000 \quad 8.4356 \quad 8.8422 \quad 19.7232$
$34.9565 \quad 50.1242 \quad 61.1748 \quad 103.5170 \quad 111.1905]$

$vint = [28.8015 \quad 21.2841 \quad 0.0000 \quad 0.0000 \quad 0.0000$
$0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000]$

$\sum_{j=1}^{n} a_j x_j = 44.2306 < b^*_{SKPDRCR}$

$tint_k = tint_3 = 3$

$x_3 = 2 < tint_3$

Let $x_3 = x_3 + 1 = 2 + 1 = 3$.

$\therefore x = [1 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$\sum_{j=1}^{n} a_j x_j = 52.9442$

Calculate u_i and v_i.

$u_i = 0$ and $v_i = \sum_{j=1}^{n} a_j x_j - b_i$, for i such that $b_i < \sum_{j=1}^{n} a_j x_j$

$\therefore u_1, u_2, u_3 = 0$

$v_1 = \sum_{j=1}^{n} a_j x_j - b_1 = 52.9442 - 15.4291 = 37.5151$

$v_2 = \sum_{j=1}^{n} a_j x_j - b_2 = 52.9442 - 22.9465 = 29.9977$

$v_3 = \sum_{j=1}^{n} a_j x_j - b_3 = 52.9442 - 52.6662 = 0.2780$

$u_i = b_i - \sum_{j=1}^{n} a_j x_j$ and $v_i = 0$, for i such that $b_i \geq \sum_{j=1}^{n} a_j x_j$

$\therefore u_4 = b_4 - \sum_{j=1}^{n} a_j x_j = 53.0728 - 52.9442 = 0.1286$

$u_5 = b_5 - \sum_{j=1}^{n} a_j x_j = 63.9538 - 52.9442 = 11.0096$

$u_6 = b_6 - \sum_{j=1}^{n} a_j x_j = 79.1871 - 52.9442 = 26.2429$

$u_7 = b_7 - \sum_{j=1}^{n} a_j x_j = 94.3548 - 52.9442 = 41.4106$

$$u_8 = b_8 - \sum_{j=1}^{n} a_j x_j = 105.4054 - 52.9442 = 52.4612$$

$$u_9 = b_9 - \sum_{j=1}^{n} a_j x_j = 147.7476 - 52.9442 = 94.8034$$

$$u_{10} = b_{10} - \sum_{j=1}^{n} a_j x_j = 155.4211 - 52.9442 = 102.4769$$

$$v_4, v_5, v_6, v_7, v_8, v_9, v_{10} = 0$$

$$f = \sum_{j=1}^{n} c_j x_j + g\sum_{i=1}^{m} p_i u_i + h\sum_{i=1}^{m} p_i v_i$$

$$= [2.4106(1) + 2.2333(2) + 3.9110(3) + 3.9655(0) + 1.3514(0)$$
$$1.3479(0) + 5.1126(0) + 9.2752(0) + 1.3257(0) + 3.1003(0)]$$
$$+ 2.8920[0.0667(0) + 0.0458(0) + 0.1753(0) + 0.0675(0.1286) + 0.0869(11.0096)$$
$$+ 0.1385(26.2429) + 0.1441(41.4106) + 0.0621(52.4612) + 0.1436(94.8034)$$
$$+ 0.0695(102.4769)]$$
$$+ 5.6602[0.0667(37.5151) + 0.0458(29.9977) + 0.1753(0.2780) + 0.0675(0)$$
$$+ 0.0869(0) + 0.1385(0) + 0.1441(0) + 0.0621(0) + 0.1436(0) + 0.0695(0)]$$
$$= 140.7765 < f_{\min}$$

$$\therefore f_{\min} = 140.7765$$

$$\therefore xint = [1 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$uint = [0.0000 \quad 0.0000 \quad 0.0000 \quad 0.1286 \quad 11.0096$$
$$26.2429 \quad 41.4106 \quad 52.4612 \quad 94.8034 \quad 102.4769]$$

$$vint = [37.5151 \quad 29.9977 \quad 0.2780 \quad 0.0000 \quad 0.0000$$
$$0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000]$$

$$\sum_{j=1}^{n} a_j x_j = 52.9442 > b^*_{SKPDRCR}$$

Integer solution is as follows.

$$x^*_{SKPDRC} = xint = [1 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$u^*_{SKPDRC} = uint = [0.0000 \quad 0.0000 \quad 0.0000 \quad 0.1286 \quad 11.0096$$
$$26.2429 \quad 41.4106 \quad 52.4612 \quad 94.8034 \quad 102.4769]$$

$$v^*_{SKPDRC} = vint = [37.5151 \quad 29.9977 \quad 0.2780 \quad 0.0000 \quad 0.0000$$
$$0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000]$$

Appendix B

Example of solving SKPCRC

Example of solving SKPCRC

$$\underset{x,u,v}{\text{Minimize}}\ f = \sum_{j=1}^{n} c_j x_j + g\int_L^U p(b)u(b)db + h\int_L^U p(b)v(b)db$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_j x_j + u(b) - v(b) = b$$

$$0 \le x_j \le t_j$$

$$\int_L^U p(b)db = 1$$

where

$g = 1.7409$

$h = 8.3729$

b is normally distributed where mean = 134.5046 and standard deviation = 7.4634.

c = [1.4623 6.4512 4.2129 8.1327 8.8142
6.9527 4.7069 8.7327 5.7908 4.4130]

a = [8.6321 8.7803 5.3610 8.3986 4.7569
2.3991 1.4834 2.1239 1.2597 0.2233]

t = [6.2884 4.2985 2.8576 3.6869 2.0542
7.1595 7.8948 9.9613 7.5732 3.8571]

$$\Delta B = 0.1\sum_{j=1}^{n} a_j t_j$$

Algorithm 1 (or Algorithm 2) for Solving SKPCRCR

Calculate c/a and $\sum a_j t_j$.

c/a = [0.1694 0.7347 0.7858 0.9683 1.8529
2.8980 3.1730 4.1116 4.5970 19.7627]

$\sum a_j t_j$ = [54.2821 92.0242 107.3438 138.3086 148.0802
165.2566 176.9677 198.1245 207.6645 208.5258]

$$\sum_{j=1}^{n} a_j t_j = 208.5258$$

$$\Delta B = 0.1\sum_{j=1}^{n} a_j t_j = 20.8526$$

$$c_1 / a_1 = 0.1694 < g$$

$$c_n / a_n \le g \int_{b\in B_1} p(b)db - h \int_{b\in B_2} p(b)db$$

$$c_n / a_n = 19.7627$$

$$B_1 = \{L \le b \le U : b \ge \sum_{j=1}^{n} a_j t_j\},\ B_2 = \{L \le b \le U : b < \sum_{j=1}^{n} a_j t_j\}$$

$$g \int_{b\in B_1} p(b)db - h \int_{b\in B_2} p(b)db = g\int_{208.5258}^{\infty} p(b)db - h\int_{-\infty}^{208.5258} p(b)db = \text{-}8.3729 < c_n / a_n$$

$$s = 1$$

$$B = s(\Delta B) = 1(20.8526) = 20.8526$$

$$\sum_{j=1}^{0} a_j t_j = 0 < B$$

$$\sum_{j=1}^{1} a_j t_j = 54.2821 > B$$

$$\therefore k = 1$$

$$cgh = c_k / a_k + h\int_{L}^{B} p(b)db - g\int_{B}^{U} p(b)db$$

$$= 0.1694 + 8.3729\int_{-\infty}^{20.8526} p(b)db - 1.7409\int_{20.8526}^{\infty} p(b)db$$

$$= 0.1694 + 8.3729(0.0000) - 1.7409(1.0000)$$

$$= \text{-}1.5715 < 0$$

$$s = 2$$

$$B = s(\Delta B) = 2(20.8526) = 41.7052$$

$$\sum_{j=1}^{0} a_j t_j = 0 < B$$

$$\sum_{j=1}^{1} a_j t_j = 54.2821 > B$$

$$\therefore k = 1$$

$$cgh = c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db$$
$$= 0.1694 + 8.3729\int_{-\infty}^{41.7052} p(b)db - 1.7409\int_{41.7052}^{\infty} p(b)db$$
$$= 0.1694 + 8.3729(0.0000) - 1.7409(1.0000)$$
$$= -1.5715 < 0$$

$s = 3$

$B = s(\Delta B) = 3(20.8526) = 62.5577$

$\sum_{j=1}^{1} a_j t_j = 54.2821 < B$

$\sum_{j=1}^{2} a_j t_j = 92.0242 > B$

$\therefore k = 2$

$$cgh = c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db$$
$$= 0.7347 + 8.3729\int_{-\infty}^{62.5577} p(b)db - 1.7409\int_{62.5577}^{\infty} p(b)db$$
$$= 0.7347 + 8.3729(0.0000) - 1.7409(1.0000)$$
$$= -1.0062 < 0$$

$s = 4$

$B = s(\Delta B) = 4(20.8526) = 83.4103$

$\sum_{j=1}^{2} a_j t_j = 92.0242 < B$

$\sum_{j=1}^{3} a_j t_j = 107.3438 > B$

$\therefore k = 3$

$$cgh = c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db$$
$$= 0.7347 + 8.3729\int_{-\infty}^{83.4103} p(b)db - 1.7409\int_{83.4103}^{\infty} p(b)db$$
$$= 0.7347 + 8.3729(0.0000) - 1.7409(1.0000)$$
$$= -1.0062 < 0$$

$s = 5$

$B = s(\Delta B) = 5(20.8526) = 104.2629$

$\sum_{j=1}^{2} a_j t_j = 92.0242 < B$

$\sum_{j=1}^{3} a_j t_j = 107.3438 > B$

$\therefore k = 3$

$$cgh = c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db$$

$$= 0.7858 + 8.3729\int_{-\infty}^{104.2629} p(b)db - 1.7409\int_{104.2629}^{\infty} p(b)db$$

$$= 0.7858 + 8.3729(0.00003) - 1.7409(0.99997)$$

$$= -0.9548 < 0$$

$s = 6$

$B = s(\Delta B) = 6(20.8526) = 125.1155$

$\sum_{j=1}^{3} a_j t_j = 107.3438 < B$

$\sum_{j=1}^{4} a_j t_j = 138.3086 > B$

$\therefore k = 4$

$$cgh = c_k / a_k + h\int_L^B p(b)db - g\int_B^U p(b)db$$

$$= 0.9683 + 8.3729\int_{-\infty}^{125.1155} p(b)db - 1.7409\int_{125.1155}^{\infty} p(b)db$$

$$= 0.9683 + 8.3729(0.1042) - 1.7409(0.8958)$$

$$= 0.2813 > 0$$

Let $B_1 = B - \Delta B$ and $B_2 = B$

$\therefore B_1 = 125.1155 - 20.8526 = 104.2629$

$B_2 = 125.1155$

$$\sum_{j=1}^{2} a_j t_j = 92.0242 < B_1$$

$$\sum_{j=1}^{3} a_j t_j = 107.3438 > B_1$$

$$\therefore k_1 = 3$$

$$f(B_1) = \sum_{j=1}^{k_1-1} c_j t_j + c_{k_1}((B_1 - \sum_{j=1}^{k_1-1} a_j t_j)/a_{k_1}) + h\int_L^{B_1} p(b)(B_1 - b)\,db + g\int_{B_1}^{U} p(b)(b - B_1)db$$

$$= \sum_{j=1}^{3-1} c_j t_j + c_3((B_1 - \sum_{j=1}^{3-1} a_j t_j)/a_3) + h\int_L^{B_1} p(b)(B_1 - b)\,db + g\int_{B_1}^{U} p(b)(b - B_1)db$$

$$= 1.4623(6.2884) + 6.4512(4.2985) + \frac{4.2129(104.2629 - 92.0242)}{5.3610}$$

$$+ 8.3729\int_{-\infty}^{104.2629} p(b)(104.2629 - b)\,db + 1.7409\int_{104.2629}^{\infty} p(b)(b - 104.2629)db$$

$$= 36.9260 + 9.6177 + 0.0003 + 52.6479$$

$$= 99.1919$$

$$\sum_{j=1}^{3} a_j t_j = 107.3438 < B_2$$

$$\sum_{j=1}^{4} a_j t_j = 138.3086 > B_2$$

$$\therefore k_2 = 4$$

$$f(B_2) = \sum_{j=1}^{k_2-1} c_j t_j + c_{k_2}((B_2 - \sum_{j=1}^{k_2-1} a_j t_j)/a_{k_2}) + h\int_L^{B_2} p(b)(B_2 - b)\,db + g\int_{B_2}^{U} p(b)(b - B_2)db$$

$$= \sum_{j=1}^{4-1} c_j t_j + c_4((B_2 - \sum_{j=1}^{4-1} a_j t_j)/a_4) + h\int_L^{B_2} p(b)(B_2 - b)\,db + g\int_{B_2}^{U} p(b)(b - B_2)db$$

$$= 1.4623(6.2884) + 6.4512(4.2985) + 4.2129(2.5876)$$

$$+ \frac{8.1327(125.1155 - 107.3438)}{8.3986} + 8.3729\int_{-\infty}^{125.1155} p(b)(125.1155 - b)\,db$$

$$+ 1.7409\int_{125.1155}^{\infty} p(b)(b - 125.1155)db$$

$$= 48.9648 + 17.2090 + 3.1086 + 16.9919$$

$$= 86.2743$$

$$f(B_2) < f(B_1)$$

$\therefore f^*_{SKPCRCR} = f(B_2)$

$k^*_{SKPCRCR} = k_2 = 4$

$B^*_{SKPCRCR} = B_2 = 125.1155$

$x_j = t_j, \quad j = 1,...,k^*_{SKPCRCR} - 1$

$\therefore x_1 = 6.2884$

$x_2 = 4.2985$

$x_3 = 2.8576$

$$x_{k^*_{SKPCRCR}} = \frac{B^*_{SKPCRCR} - \sum_{j=1}^{k^*_{SKPCRCR}-1} a_j t_j}{a_{k^*_{SKPCRCR}}}$$

$$= \frac{B^*_{SKPCRCR} - \sum_{j=1}^{4-1} a_j t_j}{a_4}$$

$$\therefore x_4 = \frac{125.1155 - 107.3438}{8.3986}$$

$$= 2.1160$$

$x_j = 0, \quad j = k^*_{SKPCRCR} + 1,...,n$

$\therefore x_5, x_6, x_7, x_8, x_9, x_{10} = 0$

$\therefore x^*_{SKPCRCR} = [6.2884 \quad 4.2985 \quad 2.8576 \quad 2.1160 \quad 0.0000$

$0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000 \quad 0.0000]$

Proposed Heuristic for SKPCRC by Using Algorithm 1 (or Algorithm 2) for Solving the Relaxed Problem

$f_{\min} = \infty$

$B^*_{SKPCRCR} = 125.1155$

$k = k^*_{SKPDRCR} = 4$

$tint = \lfloor t \rfloor = [6 \quad 4 \quad 2 \quad 3 \quad 2 \quad 7 \quad 7 \quad 9 \quad 7 \quad 3]$

$x = \lfloor x^*_{SKPCRCR} \rfloor = [6 \quad 4 \quad 2 \quad 2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$$\sum_{j=1}^{n} a_j x_j = 114.4330$$

$$f = \sum_{j=1}^{n} c_j x_j + h \int_{L}^{\sum_{j=1}^{n} a_j x_j} p(b)\left(\sum_{j=1}^{n} a_j x_j - b\right) db + g \int_{\sum_{j=1}^{n} a_j x_j}^{U} p(b)\left(b - \sum_{j=1}^{n} a_j x_j\right) db$$

$$= 1.4623(6) + 6.4512(4) + 4.2129(2) + 8.1327(2) + 8.8142(0)$$
$$+ 6.9527(0) + 4.7069(0) + 8.7327(0) + 5.7908(0) + 4.4130(0)$$
$$+ 8.3729 \int_{-\infty}^{114.4330} p(b)(114.4330 - b)\, db + 1.7409 \int_{114.4330}^{\infty} p(b)(b - 114.4330) db$$
$$= 59.2698 + 0.0686 + 34.9569$$
$$= 94.2953 < f_{\min}$$

$$\therefore f_{\min} = 94.2953$$

$$\therefore xint = [6 \quad 4 \quad 2 \quad 2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\sum_{j=1}^{n} a_j x_j = 114.4330 < B^{*}_{SKPCRCR}$$

$$tint_k = tint_4 = 3$$

$$x_4 = 2 < tint_4$$

Let $x_4 = x_4 + 1 = 2 + 1 = 3$.

$$\therefore x = [6 \quad 4 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\sum_{j=1}^{n} a_j x_j = 122.8316$$

$$f = \sum_{j=1}^{n} c_j x_j + h \int_{L}^{\sum_{j=1}^{n} a_j x_j} p(b)\left(\sum_{j=1}^{n} a_j x_j - b\right) db + g \int_{\sum_{j=1}^{n} a_j x_j}^{U} p(b)\left(b - \sum_{j=1}^{n} a_j x_j\right) db$$

$$= 1.4623(6) + 6.4512(4) + 4.2129(2) + 8.1327(3) + 8.8142(0)$$
$$+ 6.9527(0) + 4.7069(0) + 8.7327(0) + 5.7908(0) + 4.4130(0)$$
$$+ 8.3729 \int_{-\infty}^{122.8316} p(b)(122.8316 - b)\, db + 1.7409 \int_{122.8316}^{\infty} p(b)(b - 122.8316) db$$
$$= 67.4025 + 1.5801 + 20.6501$$
$$= 89.6327 < f_{\min}$$

$$\therefore f_{\min} = 89.6327$$

$$\therefore xint = [6 \quad 4 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$\sum_{j=1}^{n} a_j x_j = 122.8316 < B^*_{SKPCRCR}$

$tint_k = tint_4 = 3$

$x_4 = 3 = tint_4$

Let $x_{4+1} = x_5 = 1$.

$\therefore x = [6 \quad 4 \quad 2 \quad 3 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$\sum_{j=1}^{n} a_j x_j = 127.5885$

$$f = \sum_{j=1}^{n} c_j x_j + h \int_{L}^{\sum_{j=1}^{n} a_j x_j} p(b)\left(\sum_{j=1}^{n} a_j x_j - b\right) db + g \int_{\sum_{j=1}^{n} a_j x_j}^{U} p(b)\left(b - \sum_{j=1}^{n} a_j x_j\right) db$$

$$= 1.4623(6) + 6.4512(4) + 4.2129(2) + 8.1327(3) + 8.8142(1)$$
$$+ 6.9527(0) + 4.7069(0) + 8.7327(0) + 5.7908(0) + 4.4130(0)$$
$$+ 8.3729 \int_{-\infty}^{127.5885} p(b)(127.5885 - b)\, db + 1.7409 \int_{127.5885}^{\infty} p(b)(b - 127.5885) db$$
$$= 76.2167 + 5.9751 + 13.2826$$
$$= 95.4744 > f_{\min}$$

Integer solution is as follows.

$x^*_{SKPCRC} = xint = [6 \quad 4 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$f^*_{SKPCRC} = f_{\min} = 89.6327$

Algorithm 3 for Solving SKPCRCR

Let

$c_{n+1} = g = 1.7409$,

$a_{n+1} = 1.0000$, and

$t_{n+1} = \infty$.

$\therefore c = [1.4623 \quad 6.4512 \quad 4.2129 \quad 8.1327 \quad 8.8142$
$6.9527 \quad 4.7069 \quad 8.7327 \quad 5.7908 \quad 4.4130$
$1.7409]$

$\therefore a = [8.6321 \quad 8.7803 \quad 5.3610 \quad 8.3986 \quad 4.7569$
$2.3991 \quad 1.4834 \quad 2.1239 \quad 1.2597 \quad 0.2233$
$1.0000]$

$\therefore t = [6.2884 \quad 4.2985 \quad 2.8576 \quad 3.6869 \quad 2.0542$
$7.1595 \quad 7.8948 \quad 9.9613 \quad 7.5732 \quad 3.8571$
$\infty]$

$c/a = [0.1694 \quad 0.7347 \quad 0.7858 \quad 0.9683 \quad 1.8529$
$2.8980 \quad 3.1730 \quad 4.1116 \quad 4.5970 \quad 19.7627$
$1.7409]$

$cs/as = [0.1694 \quad 0.7347 \quad 0.7858 \quad 0.9683 \quad 1.7409$
$1.8529 \quad 2.8980 \quad 3.1730 \quad 4.1116 \quad 4.5970$
$19.7627]$

$cs = [1.4623 \quad 6.4512 \quad 4.2129 \quad 8.1327 \quad 1.7409$
$8.8142 \quad 6.9527 \quad 4.7069 \quad 8.7327 \quad 5.7908$
$4.4130]$

$as = [8.6321 \quad 8.7803 \quad 5.3610 \quad 8.3986 \quad 1.0000$
$4.7569 \quad 2.3991 \quad 1.4834 \quad 2.1239 \quad 1.2597$
$0.2233]$

$ts = [6.2884 \quad 4.2985 \quad 2.8576 \quad 3.6869 \quad \infty$
$2.0542 \quad 7.1595 \quad 7.8948 \quad 9.9613 \quad 7.5732$
$3.8571]$

$\sum as_j ts_j = [54.2821 \quad 92.0242 \quad 107.3438 \quad 138.3086 \quad \infty$
$\infty \quad \infty \quad \infty \quad \infty \quad \infty$
$\infty]$

$tr = 1$

$b_1 = 139.6543$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 < b_1$

$\sum_{j=1}^{5} as_j ts_j = \infty > b_1$

$\therefore k^* = 5$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_4 = 3.6869$

$xs_j = (b_1 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_5 = (139.6543 - 138.3086)/1$

$= 1.3457$

$xs_j = 0$, for $j = k^* + 1,...,n+1$

$\therefore xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 3.6869 1.3457
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.6869 0
0 0 0 0 0]

$tr = 2$

$b_2 = 140.5919$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 < b_2$

$\sum_{j=1}^{5} as_j ts_j = \infty > b_2$

$\therefore k^* = 5$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_4 = 3.6869$

$xs_j = (b_2 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_5 = (140.5919 - 138.3086)/1$

$= 2.2833$

$xs_j = 0, \text{ for } j = k^* + 1, ..., n+1$

$\therefore xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 3.6869 2.2833
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.6869 0
0 0 0 0 0]

$tr = 3$

$b_3 = 139.8179$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 < b_3$

$\sum_{j=1}^{5} as_j ts_j = \infty > b_3$

$\therefore k^* = 5$

$xs_j = ts_j, \text{ for } j = 1, ..., k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_4 = 3.6869$

$xs_j = (b_3 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_5 = (139.8179 - 138.3086)/1$

$= 1.5093$

$xs_j = 0, \text{ for } j = k^* + 1, ..., n+1$

$\therefore xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 3.6869 1.5093
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.6869 0
0 0 0 0 0]

$tr = 4$

$b_4 = 144.1343$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 < b_4$

$\sum_{j=1}^{5} as_j ts_j = \infty > b_4$

$\therefore k^* = 5$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_4 = 3.6869$

$xs_j = (b_4 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_5 = (144.1343 - 138.3086)/1$

$= 5.8257$

$xs_j = 0$, for $j = k^* + 1,...,n+1$

$\therefore xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 3.6869 5.8257
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.6869 0
0 0 0 0 0]

$tr = 5$

$b_5 = 125.5302$

$\sum_{j=1}^{3} as_j ts_j = 107.3438 < b_5$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 > b_5$

$\therefore k^* = 4$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_j = (b_5 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_4 = (125.5302 - 107.3438) / 8.3986$

$= 2.1654$

$xs_j = 0$, for $j = k^* + 1,...,n+1$

$\therefore xs_5, xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 2.1654 0
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 2.1654 0
0 0 0 0 0]

$tr = 6$

$b_6 = 134.3569$

$\sum_{j=1}^{3} as_j ts_j = 107.3438 < b_6$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 > b_6$

$\therefore k^* = 4$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_j = (b_6 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_4 = (134.3569 - 107.3438) / 8.3986$

$= 3.2164$

$xs_j = 0$, for $j = k^* + 1,...,n+1$

$\therefore xs_5, xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 3.2164 0
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.2164 0
0 0 0 0 0]

$tr = 7$

$b_7 = 139.4946$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 < b_7$

$\sum_{j=1}^{5} as_j ts_j = \infty > b_7$

$\therefore k^* = 5$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_4 = 3.6869$

$$xs_j = (b_7 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$$

$$\therefore xs_5 = (139.4946 - 138.3086)/1$$

$$= 1.1860$$

$$xs_j = 0, \text{ for } j = k^* + 1, ..., n+1$$

$$\therefore xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$$

$xs =$ [6.2884 4.2985 2.8576 3.6869 1.1860
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.6869 0
0 0 0 0 0]

$$tr = 8$$

$$b_8 = 133.3350$$

$$\sum_{j=1}^{3} as_j ts_j = 107.3438 < b_8$$

$$\sum_{j=1}^{4} as_j ts_j = 138.3086 > b_8$$

$$\therefore k^* = 4$$

$$xs_j = ts_j, \text{ for } j = 1, ..., k^* - 1$$

$$\therefore xs_1 = 6.2884$$

$$xs_2 = 4.2958$$

$$xs_3 = 2.8576$$

$$xs_j = (b_8 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$$

$$\therefore xs_4 = (133.3350 - 107.3438)/8.3986$$

$$= 3.0947$$

$$xs_j = 0, \text{ for } j = k^* + 1, ..., n+1$$

$$\therefore xs_5, xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$$

$xs =$ [6.2884 4.2985 2.8576 3.0947 0
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.0947 0
0 0 0 0 0]

$tr = 9$

$b_9 = 122.5327$

$\sum_{j=1}^{3} as_j ts_j = 107.3438 < b_9$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 > b_9$

$\therefore k^* = 4$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_j = (b_9 - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_4 = (122.5327 - 107.3438) / 8.3986$

$= 1.8085$

$xs_j = 0$, for $j = k^* + 1,...,n+1$

$\therefore xs_5, xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 1.8085 0
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 1.8085 0
0 0 0 0 0]

$tr = 10$

$b_{10} = 143.3923$

$\sum_{j=1}^{4} as_j ts_j = 138.3086 < b_{10}$

$\sum_{j=1}^{5} as_j ts_j = \infty > b_{10}$

$\therefore k^* = 5$

$xs_j = ts_j$, for $j = 1,...,k^* - 1$

$\therefore xs_1 = 6.2884$

$xs_2 = 4.2958$

$xs_3 = 2.8576$

$xs_4 = 3.6869$

$xs_j = (b_{10} - \sum_{j=1}^{k^*-1} as_j ts_j) / as_{k^*}$

$\therefore xs_5 = (143.3923 - 138.3086)/1$

$= 5.0837$

$xs_j = 0$, for $j = k^* + 1,...,n+1$

$\therefore xs_6, xs_7, xs_8, xs_9, xs_{10} = 0$

$xs =$ [6.2884 4.2985 2.8576 3.6869 5.0837
0 0 0 0 0
0]

$x =$ [6.2884 4.2985 2.8576 3.6869 0
0 0 0 0 0]

$x^*_{SKPCRCR} = \overline{x_j} =$ [6.2884 4.2985 2.8576 3.2406 0
0 0 0 0 0
0]

$k^*_{SKPCRCR} = 4$

$$B^*_{SKPCRCR} = \sum_{j=1}^{n} a_j \overline{x_j}$$

$$= 8.6321(6.2884) + 8.7803(4.2985) + 5.3610(2.8576) + 8.3986(3.2406)$$
$$+ 4.7569(0) + 2.3991(0) + 1.4834(0) + 2.1239(0) + 1.2597(0) + 0.2233(0)$$
$$= 134.5603$$

$$f^*_{SKPCRCR} = \sum_{j=1}^{n} c_j \overline{x_j} + h \int_{L}^{B^*_{SKPCRCR}} p(b)(B^*_{SKPCRCR} - b)\,db + g \int_{B^*_{SKPCRCR}}^{U} p(b)(b - B^*_{SKPCRCR})db$$

$$= 1.4623(6.2884) + 6.4512(4.2985) + 4.2129(2.8576) + 8.1327(3.2406)$$
$$+ 8.8142(0) + 6.9527(0) + 4.7069(0) + 8.7327(0) + 5.7908(0) + 4.4130(0)$$
$$+ 8.3729 \int_{-\infty}^{134.5603} p(b)(134.5603 - b)\,db$$
$$+ 1.7409 \int_{134.5603}^{\infty} p(b)(b - 134.5603)db$$
$$= 75.3196 + 25.1640 + 5.1351$$
$$= 105.6187$$

Proposed Heuristic for SKPCRC by Using Algorithm 3 for Solving the Relaxed Problem

$f_{\min} = \infty$

$B^*_{SKPCRCR} = 134.5603$

$k = k^*_{SKPDRCR} = 4$

$tint = \lfloor t \rfloor = [6 \quad 4 \quad 2 \quad 3 \quad 2 \quad 7 \quad 7 \quad 9 \quad 7 \quad 3]$

$x = \lfloor x^*_{SKPCRCR} \rfloor = [6 \quad 4 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$\sum_{j=1}^{n} a_j x_j = 122.8316$

$$f = \sum_{j=1}^{n} c_j x_j + h \int_{L}^{\sum_{j=1}^{n} a_j x_j} p(b)(\sum_{j=1}^{n} a_j x_j - b)\,db + g \int_{\sum_{j=1}^{n} a_j x_j}^{U} p(b)(b - \sum_{j=1}^{n} a_j x_j)db$$

$$= 1.4623(6) + 6.4512(4) + 4.2129(2) + 8.1327(3) + 8.8142(0)$$
$$+ 6.9527(0) + 4.7069(0) + 8.7327(0) + 5.7908(0) + 4.4130(0)$$
$$+ 8.3729 \int_{-\infty}^{122.8316} p(b)(122.8316 - b)\,db + 1.7409 \int_{122.8316}^{\infty} p(b)(b - 122.8316)db$$

$= 67.4025 + 1.5801 + 20.6501$

$= 89.6327 < f_{\min}$

$\therefore f_{\min} = 89.6327$

$\therefore xint = [6 \quad 4 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$\sum_{j=1}^{n} a_j x_j = 122.8316 < B^*_{SKPCRCR}$

$tint_k = tint_4 = 3$

$x_4 = 3 = tint_4$

Let $x_{4+1} = x_5 = 1$.

$\therefore x = [6 \quad 4 \quad 2 \quad 3 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$\sum_{j=1}^{n} a_j x_j = 127.5885$

$$f = \sum_{j=1}^{n} c_j x_j + h \int_{L}^{\sum_{j=1}^{n} a_j x_j} p(b)\left(\sum_{j=1}^{n} a_j x_j - b\right) db + g \int_{\sum_{j=1}^{n} a_j x_j}^{U} p(b)\left(b - \sum_{j=1}^{n} a_j x_j\right) db$$

$= 1.4623(6) + 6.4512(4) + 4.2129(2) + 8.1327(3) + 8.8142(1)$

$+ 6.9527(0) + 4.7069(0) + 8.7327(0) + 5.7908(0) + 4.4130(0)$

$+ 8.3729 \int_{-\infty}^{127.5885} p(b)(127.5885 - b)\, db + 1.7409 \int_{127.5885}^{\infty} p(b)(b - 127.5885) db$

$= 76.2167 + 5.9751 + 13.2826$

$= 95.4744 > f_{\min}$

Integer solution is as follows.

$x^*_{SKPCRC} = xint = [6 \quad 4 \quad 2 \quad 3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$f^*_{SKPCRC} = f_{\min} = 89.6327$