Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2013, Article ID 474852, 10 pages
http://dx.doi.org/10.1155/2013/474852

Research Article

A Memetic Lagrangian Heuristic for the 0-1 Multidimensional Knapsack Problem

Yourim Yoon[1] and Yong-Hyuk Kim[2]

[1] *Future IT R&D Laboratory, LG Electronics Umyeon R&D Campus, 38 Baumoe-ro, Seocho-gu, Seoul 137-724, Republic of Korea*
[2] *Department of Computer Science and Engineering, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 139-701, Republic of Korea*

Correspondence should be addressed to Yong-Hyuk Kim; yhdfly@kw.ac.kr

Received 9 January 2013; Accepted 23 April 2013

Academic Editor: Xiaohui Liu

We present a new evolutionary algorithm to solve the 0-1 multidimensional knapsack problem. We tackle the problem using duality concept, differently from traditional approaches. Our method is based on Lagrangian relaxation. Lagrange multipliers transform the problem, keeping the optimality as well as decreasing the complexity. However, it is not easy to find Lagrange multipliers nearest to the capacity constraints of the problem. Through empirical investigation of Lagrangian space, we can see the potentiality of using a memetic algorithm. So we use a memetic algorithm to find the optimal Lagrange multipliers. We show the efficiency of the proposed method by the experiments on well-known benchmark data.

1. Introduction

The knapsack problems have a number of applications in various fields, for example, cryptography, economy, network, and so forth. The 0-1 multidimensional knapsack problem (0-1MKP) is an NP-hard problem, but not strongly NP-hard [1]. It can be considered as an extended version of the well-known 0-1 knapsack problem (0-1KP). In the 0-1KP, given a set of *objects*, each object that can go into the *knapsack* has a *size* and a *profit*. The knapsack has a certain capacity for size. The objective is to find an assignment that maximizes the total profit not exceeding the given capacity. In the case of the 0-1MKP, the number of capacity constraints is more than one. For example, the constraints can be a *weight* besides a size. Naturally, the 0-1MKP is a generalized version of the 0-1KP.

Let n and m be the numbers of objects and capacity constraints, respectively. Each object i has a profit v_i, and, for each constraint j, a capacity consumption value w_{ji}. Each constraint j has a capacity b_j. Then, we formally define the 0-1MKP as follows:

$$\begin{aligned} &\text{maximize} && \mathbf{v}^T\mathbf{x} \\ &\text{subject to} && W\mathbf{x} \le \mathbf{b}, \quad \mathbf{x} \in \{0,1\}^n, \end{aligned} \tag{1}$$

where $\mathbf{v} = (v_i)$ and $\mathbf{x} = (x_i)$ are n-dimensional column vectors, $W = (w_{ji})$ is an $m \times n$ matrix, $\mathbf{b} = (b_j)$ is an m-dimensional column vector, and T means the transpose of a matrix or a column vector. W, $\mathbf{b}$, and $\mathbf{v}$ are given, and each element of them is a nonnegative integer. In brief, the objective of the 0-1MKP is to find a binary vector $\mathbf{x}$ which maximizes the weighted sum $\mathbf{v}^T\mathbf{x}$ satisfying m linear constraints $W\mathbf{x} \le \mathbf{b}$.

For the knapsack problem with *only one* constraint, there have been a number of researches about efficient approximation algorithm to find a near-optimal solution. In this paper, we are interested in the problem with more than one constraint, that is, the multidimensional knapsack problem. In [2, 3] among others, the exact algorithms for 0-1MKP have been introduced. Heuristic approaches for 0-1MKP have also been extensively studied in the past [4–13]. Also, a number of evolutionary algorithms to solve the problem have been proposed [6, 14–19]. A number of methods for the 0-1 bi-knapsack problem, which is a particular case of 0-1MKP, have also been proposed. The reader is referred to [20–22] for deep surveys of 0-1MKP.

However, most researches directly deal with the discrete search space. In this paper, we transform the search space of the problem into a real space instead of directly managing the original discrete space. The 0-1MKP is the optimization problem with multiple constraints. We transform the problem using the multiple Lagrange multipliers. However, we have a lot of limitations since the domain is not continuous but discrete. Lagrangian heuristics have been mainly used to get good upper bounds of the integer problems by the *duality*. To get good upper bounds, a number of researchers have studied dual solvings using Lagrangian duality, surrogate duality, and composite duality, and there was a recent study using cutting plane method for 0-1MKP. However, in this paper, we focus on finding good lower bounds, that is, good feasible solutions.

There have been a number of studies about Lagrangian method for discrete problems. There were also a few methods that used Lagrange multipliers for 0-1MKP. Typically, most of them found just good upper bounds to be used in a *branch-and-bound* algorithm by dualizing constraints and hence could not find a feasible solution directly. To the best of the author's knowledge, there have been only two Lagrangian methods to find lower bounds (feasible solutions). One is the constructive heuristic (CONS) proposed by Magazine and Oguz [10], and the other is its improvement using randomization (R-CONS) [13, 19]. There was also a method called LM-GA that improved the performance by combining CONS with genetic algorithms [23]. LM-GA used the real-valued weight-codings to make a variant of the original problem and then applied CONS (see Section 2.2 for details). LM-GA provided a new viewpoint to solve 0-1MKP, but it just used CONS for fitness evaluation and did not give any contribution in the aspect of Lagrangian theory.

In this paper, we present a local improvement heuristic for optimizing Lagrange multipliers. However, it is not easy to obtain good solutions by just using the heuristic itself. Through empirical investigation of Lagrangian space, we devise a novel memetic Lagrangian heuristic combined with genetic algorithms. The remainder of this paper is organized as follows. In Section 2, we present our memetic Lagrangian method together with some literature survey. In Section 3, we give our experimental results on well-known benchmark data, and we make conclusions in Section 4.

2. Lagrangian Optimization

2.1. Preliminaries. The 0-1MKP is a maximization problem with constraints. It is possible to transform the original optimization problem into the following problem using Lagrange multipliers:

$$\begin{aligned} &\text{maximize} \quad \left\{\mathbf{v}^T\mathbf{x} - \langle \boldsymbol{\lambda}, W\mathbf{x} - \mathbf{b}\rangle\right\} \\ &\text{subject to} \quad \mathbf{x} \in \{0,1\}^n. \end{aligned} \tag{2}$$

It is easy to find the maximum of the transformed problem using the following formula:

$$\mathbf{v}^T\mathbf{x} - \langle \boldsymbol{\lambda}, W\mathbf{x} - \mathbf{b}\rangle = \sum_{i=1}^{n} x_i \left(v_i - \sum_{j=1}^{m} \lambda_j w_{ji} \right) + \langle \boldsymbol{\lambda}, \mathbf{b}\rangle. \tag{3}$$

To maximize the above formula for the fixed $\boldsymbol{\lambda}$, we have to set x_i to be 1 only if $v_i > \sum_{j=1}^{m} \lambda_j w_{ji}$ for each i. Since each v_i does not have an effect on the others, getting the maximum is fairly easy. Since this algorithm computes just $\sum_{j=1}^{m} \lambda_j w_{ji}$ for each i, its time complexity becomes $O(mn)$.

If we only find out $\boldsymbol{\lambda}$ for the problem, we get the optimal solution of the 0-1MKP in polynomial time. We may have the problem that such $\boldsymbol{\lambda}$ never exists or it is difficult to find it although it exists. However, this method is not entirely useless. For arbitrary $\boldsymbol{\lambda}$, let the vector $\mathbf{x}$ which achieves the maximum in the above formula be $\mathbf{x}^*$. Since $\boldsymbol{\lambda}$ is chosen arbitrarily, we do not guarantee that $\mathbf{x}^*$ satisfies the constraints of the original problem. Nevertheless, letting the capacity be $\mathbf{b}^* = W\mathbf{x}^*$ instead of $\mathbf{b}$ makes $\mathbf{x}^*$ be the optimal solution by the following proposition [13, 19]. We call this procedure *Lagrangian method for the 0-1MKP (LMMKP)*.

Proposition 1. *The vector* $\mathbf{x}^*$ *obtained by applying LMMKP with given* $\boldsymbol{\lambda}$ *is the maximizer of the following problem:*

$$\begin{aligned} &\textit{maximize} \quad \mathbf{v}^T\mathbf{x} \\ &\textit{subject to} \quad W\mathbf{x} \le \mathbf{b}^*, \quad \mathbf{x} \in \{0,1\}^n. \end{aligned} \tag{4}$$

In particular, in the case that λ_k is 0, the kth constraint is ignored. That is, $\mathbf{x}^*$ is the maximizer of the problems which have the capacities $\mathbf{c}$'s such that $c_k \ge b_k^*$ and $c_i = b_i^*$ for all $i \ne k$. In general, the following proposition [19] holds.

Proposition 2. *In particular, if LMMKP is applied with* $\boldsymbol{\lambda}$ *such that* $\lambda_{k_1} = 0$, $\lambda_{k_2} = 0, \ldots,$ *and* $\lambda_{k_r} = 0$, *replacing the capacity* $\mathbf{b}^*$ *by* $\mathbf{c}$ *such that*

$$\begin{aligned} &c_i = b_i^*, \quad \textit{if } i \ne k_j \ \forall j, \\ &c_i \ge b_i^*, \quad \textit{otherwise} \end{aligned} \tag{5}$$

in Proposition 1 makes the proposition still hold.

Instead of finding the optimal solution of the original 0-1MKP directly, we consider the problem of finding $\boldsymbol{\lambda}$ corresponding to given constraints. That is, we transform the problem of dealing with n-dimensional binary vector $\mathbf{x}$ into the one of dealing with m-dimensional real vector $\boldsymbol{\lambda}$. If there are Lagrange multipliers corresponding to given constraints and we find them, we easily get the optimal solution of the 0-1MKP. Otherwise we try to get the solution close to the optimum by devoting to find Lagrange multipliers which satisfy given constraints and are nearest to them.

2.2. Prior Work. In this subsection, we briefly examine existing Lagrangian heuristics for discrete optimization problems. Coping with nondifferentiability of the Lagrangian led to the last technical development: *subgradient algorithm*. Subgradient algorithm is a fundamentally simple procedure. Typically, the subgradient algorithm has been used as a technique for generating good upper bounds for branch-and-bound methods, where it is known as *Lagrangian relaxation.* The reader is referred to [24] for the deep survey of Lagrangian relaxation. At each iteration of the subgradient algorithm, one takes a step from the present Lagrange multiplier in the direction opposite to a subgradient, which is the direction of

$(\mathbf{b}^* - \mathbf{b})$, where $\mathbf{b}^*$ is the capacity obtained by LMMKP, and $\mathbf{b}$ is the original capacity.

The only previous attempt to find lower bounds using Lagrangian method is CONS [10]; however, CONS without hybridization with other metaheuristics could not show satisfactory results. LM-GA by [23] obtained better results by the hybridization of weight-coded genetic algorithm and CONS. In LM-GA, a candidate solution is represented by a vector $(w_1, w_2, \dots, w_n)$ of weights. Weight w_i is associated with object i. Each profit p_i is modified by applying several biasing techniques with these weights, that is, we can obtain a modified problem instance P' which has the same constraints as those of the original problem instance but has a different object function. And then, solutions for this modified problem instance are obtained by applying a *decoding heuristic*. In particular, LM-GA used CONS as a decoding heuristic. The feasible solutions for the modified problem instance are also feasible for the original problem instance since they satisfy the same constraints. So, weight-coding does not need an explicit repairing algorithm.

The proposed heuristic is different from LM-GA in that it improves CONS itself by using properties of Lagrange multipliers, but LM-GA just uses CONS as evaluation function. Lagrange multipliers in the proposed heuristic can move to more diverse directions than CONS because of its random factor.

2.3. Randomized Constructive Heuristic. Yoon et al. [13, 19] proposed a randomized constructive heuristic (R-CONS) as an improved variant of CONS. First, $\boldsymbol{\lambda}$ is set to be $\mathbf{0}$. Consequently, x_i becomes 1 for each $v_i > 0$. It means that all positive-valued objects are put in the knapsack and so almost all constraints are violated. If $\boldsymbol{\lambda}$ is increased, some objects become taken out. We increase $\boldsymbol{\lambda}$ adequately for only one object to be taken out. We change only one Lagrange multiplier at a time. We *randomly* choose one number k and change λ_k.

Reconsider (3). Making $(v_i - \sum_{j=1}^{m} \lambda_j w_{ji})$ be negative by increasing λ_k let $x_i = 0$ by LMMKP. For each object i such that $x_i = 1$, let α_i be the increment of λ_k to make x_i be 0. Then, $(v_i - \sum_{j=1}^{m} \lambda_j w_{ji} - \alpha_i w_{ki})$ has to be negative. That is, if we increase λ_k by α_i such that $\alpha_i > (v_i - \sum_{j=1}^{m} \lambda_j w_{ji})/w_{ki}$, the ith object is taken out. So, if we just change λ_k to $\lambda_k + \min_i \alpha_i$, leave λ_i as it is for $i \neq k$, and apply LMMKP again, exactly one object is taken out. We take out objects one by one in this way and stop this procedure if every constraint is satisfied.

Algorithm 1 shows the pseudo code of R-CONS. The number of operations to take out the object is at most n, and computing α_i for each object i takes $O(m)$ time. Hence, the total time complexity becomes $O(n^2 m)$.

2.4. Local Improvement Heuristic. Our goal is to improve a real vector $\boldsymbol{\lambda}$ obtained by R-CONS whose corresponding capacity is quite close to the capacity of the given problem instance. To devise a local improvement heuristic, we exploited the following proposition [13, 19].

Proposition 3. *Suppose that $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ correspond to $\{\mathbf{x}, \mathbf{b}\}$ and $\{\mathbf{x}', \mathbf{b}'\}$ by the LMMKP, respectively. Let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_m)$ and $\boldsymbol{\lambda}' = (\lambda'_1, \lambda'_2, \dots, \lambda'_m)$, where $\lambda_i = \lambda'_i$ for $i \neq k$ and $\lambda_k \neq \lambda'_k$. Then, if $\lambda_k < \lambda'_k$, $b_k \geq b'_k$, and if $\lambda_k > \lambda'_k$, $b_k \leq b'_k$.*

Let $\mathbf{b}$ be the capacity of the given problem instance and let $\mathbf{b}^*$ be the capacity obtained by LMMKP with $\boldsymbol{\lambda}$. By the above theorem, if $b^*_k > b_k$, choosing $\boldsymbol{\lambda}' = (\lambda_1, \dots, \lambda'_k, \dots, \lambda_m)$ such that $\lambda'_k > \lambda_k$ and applying LMMKP with $\boldsymbol{\lambda}'$ makes the value of b^*_k smaller. It makes the kth constraint satisfied or the exceeded amount for the kth capacity decreased. Of course, another constraint may become violated by this operation. Also, which λ_k to be changed is at issue in the case that several constraints are not satisfied. Hence, it is necessary to set efficient rules about which λ_k to be changed and how much to change it. If good rules are made, we can find out better Lagrange multipliers than randomly generated ones quickly.

We selected the method that chooses a random number $k (\leq m)$ and increases or decreases the value of λ_k by the above theorem iteratively. In each iteration, a capacity vector $\mathbf{b}^*$ is obtained by applying LMMKP with $\boldsymbol{\lambda}$. If $\mathbf{b}^* \leq \mathbf{b}$, all constraints are satisfied and hence the best solution is updated. Since the possibility to find a better capacity exists, the algorithm does not stop here. Instead, it chooses a random number k and decreases the value of λ_k. If $\mathbf{b}^* \nleq \mathbf{b}$, we focus on satisfying constraints preferentially. For this, we choose a random number k among the numbers such that their constraints are not satisfied and increase λ_k hoping the kth value of corresponding capacity to be decreased and then the kth constraint to be satisfied. We set the amount of λ_k's change to be the fixed value δ.

Most Lagrangian heuristics for discrete problems have focused on obtaining good upper bounds, but this algorithm is distinguished in that it primarily pursues finding feasible solutions. Algorithm 2 shows the pseudo code of this local improvement heuristic. It takes $O(nmN)$ time, where N is the number of iterations.

The direction by our local improvement heuristic looks similar to that by the subgradient algorithm described in Section 2.2, but the main difference lies in that, in each iteration, the subgradient algorithm changes all coordinate values by the subgradient direction but our local improvement heuristic changes only one coordinate value. Consequently, this could make our local improvement heuristic find lower bounds more easily than subgradient algorithm usually producing upper bounds.

2.5. Investigation of the Lagrangian Space. The structure of the problem space is an important factor to indicate the problem difficulty, and the analysis of the structure helps efficient search in the problem space [25–27]. Recently Puchinger et al. [28] gave some insight into the solution structure of 0-1MKP. In this subsection, we conduct some experiments and get some insight into the global structure of the 0-1MKP space.

In Section 2.1, we showed that there is a correspondence between binary solution vector and Lagrange multiplier

```
R-CONS(0-1MKP instance)
{
    $\boldsymbol{\lambda} \leftarrow \mathbf{0}$;
    $I \leftarrow \{1, 2, \ldots, n\}$;
    do
        $k \leftarrow$ random integer in $[1, m]$;
        for $i \in I$
            $\alpha_i \leftarrow (v_i - \sum_{j=1}^{m} \lambda_j w_{ji})/w_{ki}$;
        $\lambda_k \leftarrow \lambda_k + \min_{i \in I} \alpha_i$;
        $I \leftarrow I \setminus \{\arg\min_{i \in I} \alpha_i\}$;
        $(\mathbf{x}^*, \mathbf{b}^*) = \mathrm{LMMKP}(\boldsymbol{\lambda})$;
    until $\mathbf{x}^*$ satisfies all the constraints;
    return $\boldsymbol{\lambda}$;
}
```

Algorithm 1: Randomized constructive heuristic [13, 19].

```
LocalImprovementHeuristic($\boldsymbol{\lambda}$, 0-1MKP instance)
{
    for $i \leftarrow 1$ to $N$
        $(\mathbf{x}^*, \mathbf{b}^*) \leftarrow \mathrm{LMMKP}(\boldsymbol{\lambda})$;
        $I \leftarrow \{i : b_i^* \le b_i\}$ and $J \leftarrow \{i : b_i^* > b_i\}$;
        if $I = \{1, 2, \ldots, m\}$
            update the best solution;
            choose a random element $k$ among $I$;
            $\lambda_k \leftarrow \lambda_k - \delta$;
        else
            choose a random element $k$ among $J$;
            $\lambda_k \leftarrow \lambda_k + \delta$;
    return $\boldsymbol{\lambda}$ corresponding to the best solution;
}
```

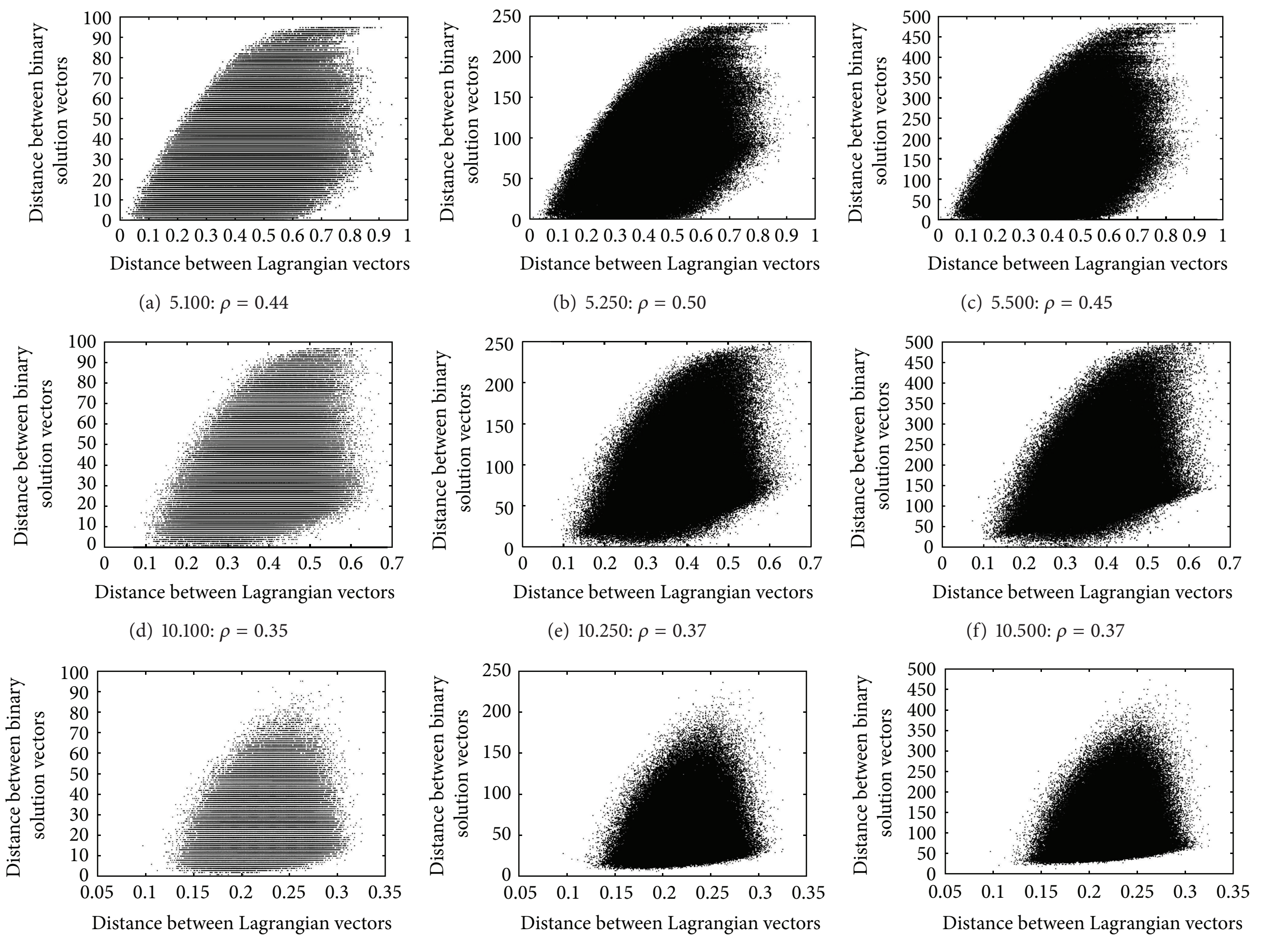
Algorithm 2: Local improvement heuristic.

vector. Strictly speaking, there cannot be a one-to-one correspondence in the technical sense of *bijection*. The binary solution vector has only binary components, so there are only countably many such vectors. But the Lagrange multipliers are real numbers, so there are uncountably many Lagrange multiplier vectors. Several multiplier vectors may correspond to the same binary solution vector. Moreover, some multiplier vectors may have multiple binary solution vectors.

Instead of directly finding an optimal binary solution, we deal with Lagrange multipliers. In this subsection, we empirically investigate the relationship between binary solution space and Lagrangian space (i.e., $\{\mathbf{x}_{\boldsymbol{\lambda}}\text{s}\}$ and $\{\boldsymbol{\lambda}\text{s}\}$).

We made experiments on nine instances ($m \cdot n$) changing the number of constraints (m) from 5 to 30 and the number of objects from 100 to 500. We chose a thousand of randomly generated Lagrange multipliers and plotted, for each pair of Lagrange multiplier vectors, the relation between the Hamming distance in binary solution space and the Euclidean distance in Lagrangian space. Figure 1 shows the plotting results. The smaller the number of constraints (m) is, the larger the Pearson correlation coefficient (ρ) is. We also made the same experiments on locally optimal Lagrange multipliers. Figure 2 shows the plotting results. Locally optimal Lagrange multipliers show much stronger correlation than randomly generated ones. They show strong positive correlation (much greater than 0.5). It means that binary solution space and locally optimal Lagrangian space are roughly isometric. The results show that both spaces have similar neighborhood structures. So this hints that it is easy to find high-quality Lagrange multipliers satisfying all the capacity constraints by using memetic algorithms on locally optimal Lagrangian space. That is, memetic algorithms can be a good choice for searching Lagrangian space directly.

2.6. Proposed Memetic Algorithm. A genetic algorithm (GA) is a problem-solving technique motivated by Darwin's theory of natural selection in evolution. A GA starts with a set of initial solutions, which is called a *population*. Each solution in the population is called a *chromosome*, which is typically represented by a linear string. This population then evolves into different populations for a number of iterations (generations). At the end, the algorithm returns the best chromosome of the population as the solution to the problem. For each iteration, the evolution proceeds in the following. Two solutions of the population are chosen based on some probability distribution. These two solutions



(a) 5.100: $\rho = 0.44$

(b) 5.250: $\rho = 0.50$

(c) 5.500: $\rho = 0.45$

(d) 10.100: $\rho = 0.35$

(e) 10.250: $\rho = 0.37$

(f) 10.500: $\rho = 0.37$

(g) 30.100: $\rho = 0.19$

(h) 30.250: $\rho = 0.19$

(i) 30.500: $\rho = 0.19$

FIGURE 1: Relationship between distances on solution space and those on Lagrangian space (among randomly generated Lagrangian vectors). $^*x$-axis: distance between Lagrangian vectors, y-axis: distance between binary solution vectors, and ρ: Pearson correlation coefficient.

are then combined through a *crossover* operator to produce an offspring. With low probability, this offspring is then modified by a *mutation* operator to introduce unexplored search space into the population, enhancing the diversity of the population. In this way, offsprings are generated and they *replace* part of or the whole population. The evolution process is repeated until a certain condition is satisfied, for example, after a fixed number of iterations. A GA that generates a considerable number of offsprings per iteration is called a *generational* GA, as opposed to a *steady-state* GA which generates only one offspring per iteration. If we apply a local improvement heuristic typically after the mutation step, the GA is called a *memetic algorithm* (MA). Algorithm 3 shows a typical generational MA.

We propose an MA for optimizing Lagrange multipliers. It conducts search using an evaluation function with penalties for violated capacity constraints. Our MA provides an alternative search method to find a good solution by optimizing m Lagrange multipliers instead of directly dealing with binary vectors with length n ($m \ll n$).

The general framework of an MA is used in our study. In the following, we describe each part of the MA.

Encoding. Each solution in the population is represented by a chromosome. Each chromosome consists of m genes corresponding to Lagrange multipliers. A real encoding is used for representing the chromosome $\boldsymbol{\lambda}$.

Initialization. The MA first creates initial chromosomes using R-CONS described in Section 2.3. We set the population size P to be 100.

Mating and Crossover. To select two parents, we use a random mating scheme. A crossover operator creates a new offspring by combining parts of the parents. We use the uniform crossover.

Mutation. After the crossover, mutation operator is applied to the offspring. We use a gene-wise mutation. After generating a random number r from 1 to m, the value of each gene is divided by r.

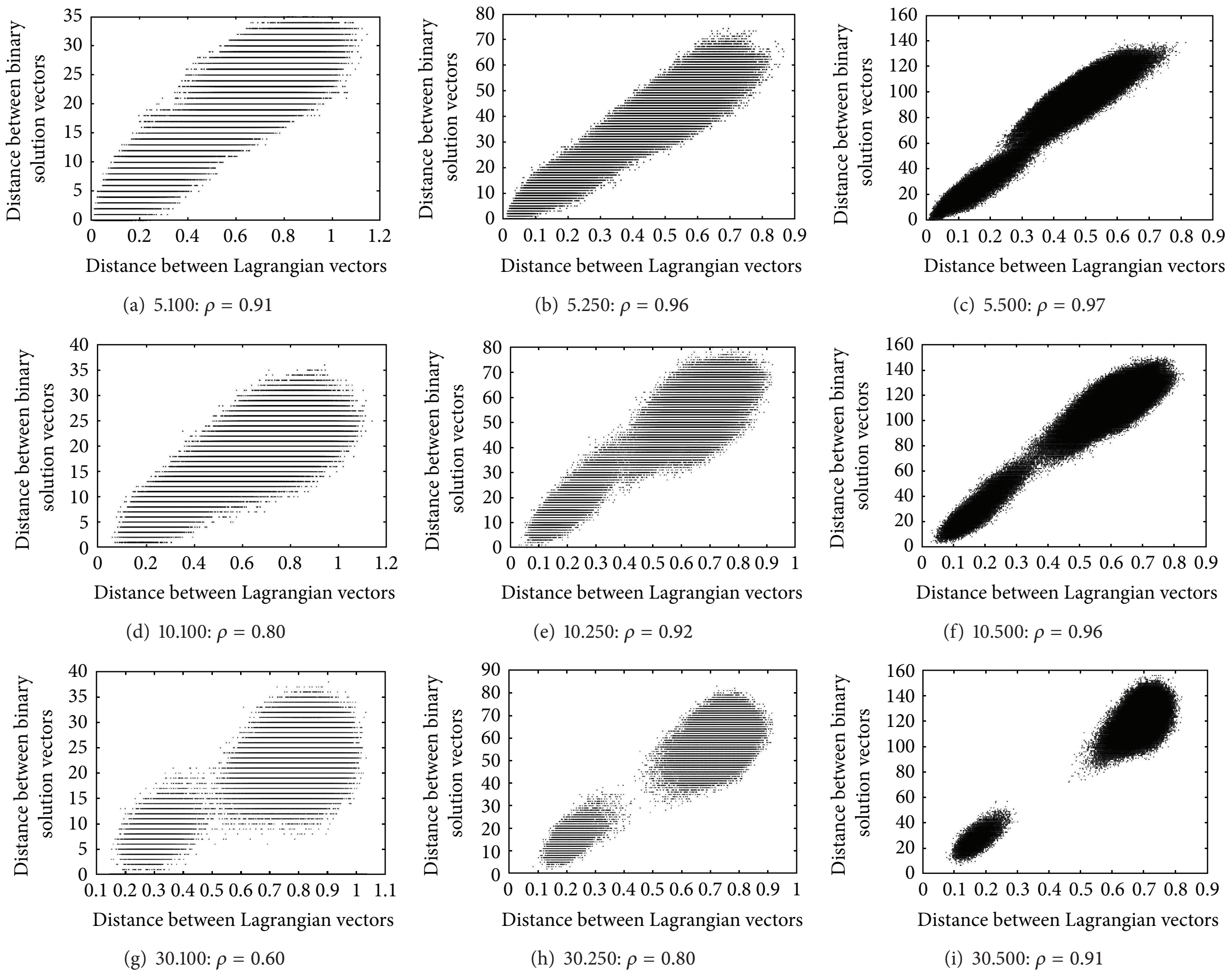


Figure 2: Relationship between distances on solution space and those on Lagrangian space (among locally optimal Lagrangian vectors). *x-axis: distance between Lagrangian vectors, y-axis: distance between binary solution vectors, and ρ: Pearson correlation coefficient.

```
create an initial population of a fixed size $P$;
do
    for $i \leftarrow 1$ to $P/2$
        choose $parent1_i$ and $parent2_i$ from population;
        $offspring_i \leftarrow$ crossover($parent1_i$, $parent2_i$);
        mutation($offspring_i$);
        apply a local improvement heuristic to $offspring_i$;
    replace(population, $\{offspring_i\}$);
until (stopping condition)
return the best individual;
```

Algorithm 3: Framework of our memetic algorithm.

Local Improvement. We use a local improvement heuristic described in Section 2.4. The number of iterations (N) is set to be 30,000. We set δ to 0.0002.

Replacement and Stopping Condition. After generating $P/2$ offspring, our MA chooses the best P individual among the total $3P/2$ ones as the population of the next generation. Our MA stops when the number of generations reaches 100.

Evaluation Function. Our evaluation function is to find a Lagrange multiplier vector $\boldsymbol{\lambda}$ that has a high fitness satisfying the capacity constraints as *much* as possible. In our MA, the following is used as the objective function to maximize, which is the function obtained by subtracting the penalty from the objective function of the 0-1MKP:

$$\mathbf{v}^T\mathbf{x}^* - \gamma \sum_{\mathbf{b}_i^* > \mathbf{b}_i} \left(\mathbf{b}_i^* - \mathbf{b}_i\right), \qquad (6)$$

Table 1: Results of local search heuristics on benchmark data.

Instance	CONS [10]	R-CONS [13, 19]			R-CONS-L		
	Result[1]	Best[2]	Ave[2]	CPU[3]	Best[2]	Ave[2]	CPU[3]
5.100–0.25	13.70	9.71	22.32	2.20	2.47	3.20	286
5.100–0.50	7.25	5.73	14.00	1.57	1.10	1.35	292
5.100–0.75	5.14	3.70	7.98	0.87	0.72	0.83	278
Average (5.100−∗)	8.70	6.38	14.77	1.55	1.43	1.79	285
5.250–0.25	6.77	7.62	17.65	13.23	0.92	1.03	703
5.250–0.50	5.27	4.94	11.27	9.24	0.49	0.56	722
5.250–0.75	3.55	3.02	6.26	5.06	0.26	0.29	679
Average (5.250−∗)	5.20	5.19	11.73	9.18	0.56	0.63	701
5.500–0.25	4.93	7.80	15.27	53.00	0.48	0.50	1414
5.500–0.50	2.65	4.18	9.45	36.56	0.20	0.22	1464
5.500–0.75	2.22	2.63	5.43	19.75	0.14	0.15	1367
Average (5.500−∗)	3.27	4.87	10.05	36.44	0.27	0.29	1415
Average (5.∗−∗)	5.72	5.48	12.18	15.72	0.75	0.91	800
10.100–0.25	15.88	11.49	22.93	3.66	2.75	6.30	536
10.100–0.50	10.01	7.75	14.76	2.63	1.47	3.68	543
10.100–0.75	6.57	3.61	8.05	1.42	0.91	2.02	532
Average (10.100−∗)	10.82	7.62	15.25	2.57	1.71	4.00	537
10.250–0.25	11.26	10.32	17.26	22.19	1.22	3.87	1321
10.250–0.50	6.49	6.46	11.51	15.65	0.58	2.07	1344
10.250–0.75	4.16	3.43	6.05	8.40	0.36	1.23	1306
Average (10.250−∗)	7.30	6.74	11.61	15.41	0.72	2.39	1323
10.500–0.25	8.27	9.59	14.91	88.30	0.64	2.89	2641
10.500–0.50	5.26	5.56	9.28	67.33	0.27	1.39	2686
10.500–0.75	3.49	3.28	5.24	32.50	0.22	0.90	2601
Average (10.500−∗)	5.67	6.14	9.81	62.71	0.37	1.73	2642
Average (10.∗−∗)	7.93	6.83	12.22	26.90	0.93	2.71	1501
30.100–0.25	16.63	11.75	22.02	9.55	5.06	9.17	1365
30.100–0.50	10.31	7.51	14.57	6.80	2.80	5.52	1374
30.100–0.75	6.60	4.20	7.97	3.79	1.64	3.22	1363
Average (30.100−∗)	11.18	7.82	14.85	6.71	3.17	5.97	1367
30.250–0.25	13.32	11.36	17.01	58.46	4.42	6.84	3364
30.250–0.50	8.19	6.71	10.87	40.51	2.36	4.11	3388
30.250–0.75	4.45	3.56	5.86	21.73	1.44	2.39	3348
Average (30.250−∗)	8.65	7.21	11.25	40.23	2.74	4.45	3366
30.500–0.25	10.34	9.39	14.02	228.56	4.00	5.94	6726
30.500–0.50	6.80	6.17	9.20	157.49	2.23	3.69	6755
30.500–0.75	3.82	3.34	4.90	83.43	1.27	2.14	6659
Average (30.500−∗)	6.99	6.30	9.37	156.49	2.50	3.92	6713
Average (30.∗−∗)	8.94	7.11	11.82	67.81	2.80	4.78	3815
Total average	7.53	6.47	12.08	36.81	1.50	2.80	2039

[1]Since CONS is a deterministic algorithm, each run always outputs the same result.
[2]Results from 1,000 runs.
[3]Total CPU seconds on Pentium III 997 MHz.

where γ is a constant which indicates the degree of penalty, and we used a fixed value 0.7.

3. Experiments

We made experiments on well-known benchmark data publicly available from the OR-Library [29], which are the same as those used in [6]. They are composed of 270 instances with 5, 10, and 30 constraints. They have different numbers of objects and different tightness ratios. The tightness ratio means α such that $b_j = \alpha \sum_{i=1}^{n} w_{ji}$ for each $j = 1, 2, \ldots, m$. The class of instances are briefly described below.

> $m.n - \alpha$: m constraints, n objects, and tightness ratio α. Each class has 10 instances.

The proposed algorithms were implemented with *gcc* compiler on a Pentium III PC (997 MHz) using Linux operating system. As the measure of performance, we

Table 2: Results of memetic Lagrangian heuristic on benchmark data.

Instance	Multistart R-CONS-L[1]		Memetic Lagrangian heuristic		Number improved/number equalled
	Result	CPU[2]	Result	CPU[2]	
5.100–0.25	2.40	1435	2.32	1466	0/0
5.100–0.50	1.08	1471	1.08	1500	0/0
5.100–0.75	0.72	1394	0.72	1431	0/0
Average (5.100−∗)	1.40	1433	1.37	1466	Total 0/0
5.250–0.25	0.92	3513	0.92	3619	0/0
5.250–0.50	0.49	3606	0.49	3748	0/0
5.250–0.75	0.26	3395	0.25	3529	0/0
Average (5.250−∗)	0.56	3505	0.56	3632	Total 0/0
5.500–0.25	0.48	7072	0.48	7124	0/0
5.500–0.50	0.20	7318	0.20	7546	0/0
5.500–0.75	0.14	6846	0.14	6985	0/0
Average (5.500−∗)	0.27	7079	0.27	7218	Total 0/0
Average (5.∗−∗)	0.74	4006	0.73	4105	Total 0/0
10.100–0.25	2.45	2679	2.27	2630	0/0
10.100–0.50	1.46	2717	1.14	2650	0/1
10.100–0.75	0.89	2657	0.69	2598	0/0
Average (10.100−∗)	1.60	2684	1.37	2626	Total 0/1
10.250–0.25	1.14	6599	0.88	6441	0/0
10.250–0.50	0.57	6718	0.45	6558	0/0
10.250–0.75	0.35	6535	0.24	6408	0-1
Average (10.250−∗)	0.69	6617	0.52	6469	Total 0/1
10.500–0.25	0.58	13196	0.50	12811	0/0
10.500–0.50	0.27	13425	0.24	13057	0/0
10.500–0.75	0.21	13006	0.14	12750	0/0
Average (10.500−∗)	0.35	13209	0.29	12873	Total 0/0
Average (10.∗−∗)	0.88	7504	0.73	7323	Total 0/2
30.100–0.25	4.92	6824	3.19	6796	0/3
30.100–0.50	2.67	6870	1.43	7052	0/3
30.100–0.75	1.56	6817	0.89	6780	0/2
Average (30.100−∗)	3.05	6837	1.84	6876	Total 0/8
30.250–0.25	4.24	16819	1.28	16799	2/2
30.250–0.50	2.23	16941	0.59	16958	0/0
30.250–0.75	1.40	16742	0.34	16837	0/0
Average (30.250−∗)	2.62	16834	0.74	16865	Total 2/2
30.500–0.25	3.81	33630	0.68	33414	2/0
30.500–0.50	2.13	33802	0.29	33737	1/0
30.500–0.75	1.24	33318	0.19	33440	0/0
Average (30.500−∗)	2.39	33583	0.39	33530	Total 3/0
Average (30.∗−∗)	2.69	19085	0.99	19090	Total 5/10
Total average	1.44	10198	0.82	10173	Total 5/12

[1]Multistart R-CONS-L returns the best result from 5,000 independent runs of R-CONS-L.
[2]Average CPU seconds on Pentium III 997 MHz.

used the percentage difference-ratio $100 \times |LP\_optimum - output|/output$ which was used in [6], where *LP_optimum* is the optimal solution of the linear programming relaxation over $\mathbb{R}$. It has a value in the range of $[0, 100]$. The smaller the value is, the smaller the difference from the optimum is.

First, we compared constructive heuristics and local improvement heuristic. Table 1 shows the results of CONS, R-CONS, and R-CONS-L, where R-CONS-L starts with a solution produced by R-CONS and locally improves it by the local improvement heuristic described in Section 2.4. We can see that R-CONS-L largely improves the results of R-CONS.

Next, to verify the effectiveness of the proposed MA, we compared the results with a *multistart* method using R-CONS-L. Multistart R-CONS-L returns the best result from

5,000 independent runs of R-CONS-L. Table 2 shows the results of multistart R-CONS-L and our MA. The proposed MA outperformed multistart R-CONS-L.

The last column in Table 2 corresponds to the numbers of improved and equalled results compared with the best results of [6]. Surprisingly, the proposed memetic algorithm could find better results than one of the state-of-the-art genetic algorithms [6] for some instances with 30 constraints (5 best over 90 instances).

4. Concluding Remarks

In this paper, we tried to find good feasible solutions by using a new memetic algorithm based on Lagrangian relaxation. To the best of the author's knowledge, it is the first trial to use a memetic algorithm when optimizing Lagrange multipliers.

The Lagrangian method guarantees optimality with the capacity constraints which may be different from those of the original problem. But it is not easy to find Lagrange multipliers that accord with all the capacity constraints. Through empirical investigation of Lagrangian space, we knew that the space of locally optimal Lagrangian vectors and that of their corresponding binary solutions are roughly isometric. Based on this fact, we applied a memetic algorithm, which searches only locally optimal Lagrangian vectors, to the transformed problem encoded by real values, that is, Lagrange multipliers. Then we could find high-quality Lagrange multipliers by the proposed memetic algorithm. We obtained a significant performance improvement over R-CONS and multistart heuristic.

In recent years, there have been researches showing better performance than Chu and Beasley [6] on the benchmark data of 0-1MKP [4, 5, 8, 9, 11–13]. Although the presented memetic algorithm did not dominate such state-of-the-art methods, this study could show the potentiality of memetic search on Lagrangian space. Because we used simple genetic operators, we believe that there is room for further improvement on our memetic search. The improvement using enhanced genetic operators tailored to real encoding, for example, [30], will be a good future work.

Acknowledgments

The present research has been conducted by the research grant of Kwangwoon University in 2013. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) and funded by the Ministry of Education, Science and Technology (2012-0001855).

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, Calif, USA, 1979.

[2] R. Mansini and M. G. Speranza, "CORAL: an exact algorithm for the multidimensional knapsack problem," *INFORMS Journal on Computing*, vol. 24, no. 3, pp. 399–415, 2012.

[3] S. Martello and P. Toth, "An exact algorithm for the two-constraint 0-1 knapsack problem," *Operations Research*, vol. 51, no. 5, pp. 826–835, 2003.

[4] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon, "A multi-level search strategy for the 0-1 multidimensional knapsack problem," *Discrete Applied Mathematics*, vol. 158, no. 2, pp. 97–109, 2010.

[5] V. Boyer, M. Elkihel, and D. El Baz, "Heuristics for the 0-1 multidimensional knapsack problem," *European Journal of Operational Research*, vol. 199, no. 3, pp. 658–664, 2009.

[6] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional Knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[7] F. Della Croce and A. Grosso, "Improved core problem based heuristics for the 0-1 multi-dimensional knapsack problem," *Computers & Operations Research*, vol. 39, no. 1, pp. 27–31, 2012.

[8] K. Fleszar and K. S. Hindi, "Fast, effective heuristics for the 0-1 multi-dimensional knapsack problem," *Computers & Operations Research*, vol. 36, no. 5, pp. 1602–1607, 2009.

[9] S. Hanafi and C. Wilbaut, "Improved convergent heuristics for the 0-1 multidimensional knapsack problem," *Annals of Operations Research*, vol. 183, no. 1, pp. 125–142, 2011.

[10] M. J. Magazine and O. Oguz, "A heuristic algorithm for the multidimensional zero-one knapsack problem," *European Journal of Operational Research*, vol. 16, no. 3, pp. 319–326, 1984.

[11] M. Vasquez and Y. Vimont, "Improved results on the 0-1 multidimensional knapsack problem," *European Journal of Operational Research*, vol. 165, no. 1, pp. 70–81, 2005.

[12] Y. Vimont, S. Boussier, and M. Vasquez, "Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem," *Journal of Combinatorial Optimization*, vol. 15, no. 2, pp. 165–178, 2008.

[13] Y. Yoon, Y.-H. Kim, and B.-R. Moon, "A theoretical and empirical investigation on the Lagrangian capacities of the 0-1 multidimensional knapsack problem," *European Journal of Operational Research*, vol. 218, no. 2, pp. 366–376, 2012.

[14] C. Cotta and J. M. Troya, "A hybrid genetic algorithm for the 0-1 multiple knapsack problem," in *Artificial Neural Networks and Genetic Algorithms*, G. D. Smith, N. C. Steele, and R. F. Albrecht, Eds., vol. 3, pp. 251–255, Springer, New york, NY, USA, 1997.

[15] J. Gottlieb, *Evolutionary algorithms for constrained optimization problems [Ph.D. thesis]*, Department of Computer Science, Technical University of Clausthal, Clausthal, Germany, 1999.

[16] S. Khuri, T. Bäck, and J. Heitkötter, "The zero/one multiple knapsack problem and genetic algorithmspages," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 188–193, ACM Press, 1994.

[17] G. R. Raidl, "Improved genetic algorithm for the multiconstrained 0-1 knapsack problem," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 207–211, May 1998.

[18] J. Thiel and S. Voss, "Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms," *INFOR*, vol. 32, no. 4, pp. 226–242, 1994.

[19] Y. Yoon, Y.-H. Kim, and B.-R. Moon, "An evolutionary Lagrangian method for the 0-1 multiple knapsack problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 629–635, June 2005.

[20] A. Fréville, "The multidimensional 0-1 knapsack problem: an overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1–21, 2004.

[21] A. Fréville and S. Hanafi, "The multidimensional 0-1 knapsack problem-bounds and computational aspects," *Annals of Operations Research*, vol. 139, no. 1, pp. 195–227, 2005.

[22] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*, Springer, Berlin, Germany, 2004.

[23] G. R. Raidl, "Weight-codings in a genetic algorithm for the multiconstraint knapsack problem," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 596–603, 1999.

[24] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, no. 1, pp. 1–18, 1981.

[25] K. D. Boese, A. B. Kahng, and S. Muddu, "A new adaptive multi-start technique for combinatorial global optimizations," *Operations Research Letters*, vol. 16, no. 2, pp. 101–113, 1994.

[26] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 184–192, 1995.

[27] Y.-H. Kim and B.-R. Moon, "Investigation of the fitness landscapes in graph bipartitioning: an empirical study," *Journal of Heuristics*, vol. 10, no. 2, pp. 111–133, 2004.

[28] J. Puchinger, G. R. Raidl, and U. Pferschy, "The multidimensional knapsack problem: structure and algorithms," *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 250–265, 2010.

[29] J. E. Beasley, "Obtaining test problems via Internet," *Journal of Global Optimization*, vol. 8, no. 4, pp. 429–433, 1996.

[30] Y. Yoon, Y.-H. Kim, A. Moraglio, and B.-R. Moon, "A theoretical and empirical study on unbiased boundary-extended crossover for real-valued representation," *Information Sciences*, vol. 183, no. 1, pp. 48–65, 2012.