Application and Assessment of the Divide-and-Conquer method on some Integer Optimization Problems

Fernando A Morales*

*Escuela de Matemáticas Universidad Nacional de Colombia, Sede Medellín
Carrera 65 # 59A–110, Bloque 43, of 106, Medellín - Colombia

Abstract

In this paper the Divide-and-Conquer method is applied and assessed to three integer optimization problems: Multidimensional Knapsack Problem (d-KP), Bin Packing Problem (BPP) and Travelling Salesman Problem (TSP). For each case, the method is introduced, together with the design of numerical experiments, in order to empirically establish its performance from both points of view: its computational time and its numerical accuracy.

Keywords: Divide-and-Conquer Method, Multidimensional Knapsack Problem, Bin Packing Problem, Traveling Salesman Problem, Monte Carlo simulations, method’s efficiency.

2010 MSC: 90C10, 68Q87

1. Introduction

Broadly speaking, the Divide-and-Conquer method for Integer Optimization problems aims to reduce the computational complexity of the problem, at the price of losing accuracy on the solutions. So far, it has been successfully introduced in [1] by Morales and Martinez for solving the 0-1 Knapsack Problem (0-1KP). Its success lies on the fact that it is a good balance between computational complexity and solution quality. It must be noted that the method does not compete with existing algorithms, it complements them. In addition, the method is ideally suited for parallel implementation, which is crucial in order to attain its full advantage. The aim of this paper is to extend the Divide-and-Conquer method and assess its performance to three Integer Optimization problems: Multidimensional Knapsack Problem (d-KP), Bin Packing Problem (BPP) and the Traveling Salesman Problem (TSP).

Throughout the problems presented here (as well as 0-1KP in [1]), the Divide-and-Conquer method consists in dividing an original integer optimization problem Π, in two subproblems, namely Π_left, Π_right, using a greedy function as the main criterion for such subdivision. The process can be iterated several times on the newly generated subproblems, creating a tree whose nodes are subproblems and whose leaves have an adequate (previously decided) size. The latter are the subproblems to be solved, either exactly or approximately by an oracle algorithm (an existing algorithm) chosen by the user according to the needs. Finally, a reassemble process is done on the aforementioned tree to recover a feasible solution for the original problem Π; this is the solution delivered by the method.

The 0-1KP has been extensively studied from many points of view, in contrast d-KP has been given little attention despite its importance. Among the heuristic methods for approximating d-KP the papers [2, 3, 4] work on greedy methods, see [5, 6] for heuristics based on the dual simplex algorithm (both pursuing polynomial time
approximations), see [7] for a constraint-relaxation based strategy, see [8] for a Lagrange multipliers approach and [9] for a genetic algorithm approach. Finally, the generalized multidemand multidimensional knapsack problem is treated heuristically in [10] using tabu search principles to detect cutting hyperplanes. As important as all these methods are, it can be seen that none of them introduces the Divide-and-Conquer principle; matter of fact, this paradigm was introduced to the 0-1KP for the first time in [1]. It follows that the strategy introduced in Section 2 is not only new (to the Author’s best knowledge) but it is complementary to any of the aforementioned methods.

The BPP ranks among the most studied problems in combinatorial optimization. Most of the efforts are focused on describing its approximation properties (see [11, 12]) or finding lower bounds for the worst case performance ratio of the proposed heuristic algorithms (see [13, 14]). It is important to stress that among the many heuristics introduced to solve BPP, the algorithm presented in [15] follows the Divide-and-Conquer strategy, although it uses a harmonic partition of the items; which is different from the partition presented in Section 3. Yet again, the analysis presented by the Authors concentrates on lower bounds determination for the worst case scenario. Furthermore, from the worst case performance point of view, the classic Best Fit and First Fit algorithms have the same behavior as a large class of packing rules. It follows that a expected performance ratio analysis is needed in order to have more insight on the algorithm’s performance; however, little attention has been given in this direction (see [16]) which is the target of our analysis here. As before the Divide-and-Conquer method combines perfectly with any of the preexisting ones, although the analysis here will be limited to its interaction with the most basic algorithms.

Finally, from the perspective of the TSP, the Divide-and-Conquer method is included in the family of heuristics reducing the size of the problem using some criteria, fixing edges [17], introducing multiple levels of reduction [18, 19] or the introduction of sub-tours from a previously known one [20]. The sub-problems are solved using a known algorithm (or oracle) whether exact [21, 22, 23] or approximate [24, 25]. Next, a merging tour algorithm [26, 27] is used to build a global tour. The present work applied to TSP in Section 4 has the structure just described.

The paper is organized as follows. In the rest of this section the notation is introduced, as well as the common criteria for designing the numerical experiments. Each of the analyzed problems has a separate section, hence Section 2 is dedicated for d-KP, Section 3 for BPP and Section 4 is devoted to TSP. In the three cases, the section starts with a brief review of the problem, continues introducing the Divide-and-Conquer method for the problem at hand and closes with the corresponding numerical experiments (description and results). Finally, Section 5 delivers the conclusions and final discussion of the work.

1.1. Notation

In this section the mathematical notation is introduced. For any natural number $N \in \mathbb{N}$, the symbol $[N] \stackrel{def}{=} \{1, 2, \ldots, N\}$ indicates the sorted set of the first $N$ natural numbers. A particularly important set is $S_N$, where $S_N$ denotes the collection of all permutations in $[N]$. For any real number $x \in \mathbb{R}$, the floor and ceiling function are given (and denoted) by $\lfloor x \rfloor \stackrel{def}{=} \max\{k : k \leq x, k \text{ integer}\}$, $\lceil x \rceil \stackrel{def}{=} \max\{k : k \geq x, k \text{ integer}\}$, respectively. Given an instance of a problem the symbols $z_{\text{DC}}$ and $z^*$ indicate the optimal solution given by the Divide-and-Conquer method and the optimal solution respectively. The analogous notation is used for $T_{\text{DC}}, T^*$ for the corresponding computational times.

1.2. Numerical Experiments Design

The numerical experiments are aimed to asses the performance of the Divided-and-Conquer method presented. Our ultimate goal is to reliably compute the expected solution’s accuracy and the computational time furnished by the method. To that end the probabilistic approach is adopted, whose construction is based on the Law of Large Numbers and the Central Limit Theorem (which is written below for the sake of completeness). The theorem 1 below assures the convergence of the random variables while the theorem 2 yields the number of necessary Bernoulli trials, in order to assure that our computed (averaged) values lie within a confidence interval.
Theorem 1 (Law of Large Numbers). Let \(( Z^{(n)} : n \in \mathbb{N} )\) be a sequence of independent, identically distributed random variables with expectation \( \mathbb{E}(Z^{(1)}) \), then
\[
\mathbb{P}\left[\frac{Z^{(1)} + Z^{(2)} + \ldots + Z^{(n)}}{n} - \mathbb{E}(Z^{(1)}) > 0\right] \rightarrow 0_{n \to \infty},
\]i.e., the sequence \(( Z^{(n)} : n \in \mathbb{N} )\) converges to \(\mu\) in the Cesàro sense.

Proof. The proof and details can be found in [28].

Theorem 2. Let \(x\) be a scalar statistical variable with mean \(\bar{x}\) and variance \(\sigma^2\).

(i) The number of Bernoulli trials necessary to get a 95\% confidence interval is given by
\[
n \overset{\text{def}}{=} \left(\frac{1.96}{0.05}\right)^2 \sigma^2.
\]

(ii) The 95\% confidence interval is given by
\[
I_x \overset{\text{def}}{=} \left[\bar{x} - 1.96 \sqrt{\frac{\sigma^2}{n}} - \bar{x} + 1.96 \sqrt{\frac{\sigma^2}{n}}\right].
\]

Proof. The proof is based on the Central Limit Theorem, see [29] for details.

2. The Multidimensional Knapsack Problem (d-KP)

In the current section the Multidimensional Knapsack Problem (d-KP) is recalled, and the Divide-and-Conquer method adjusted to this problem is presented based on a strategic choice of efficiency coefficients.

Problem 1 (Multidimensional Knapsack Problem, d-KP). Consider the problem
\[
z^* \overset{\text{def}}{=} \max \sum_{j=1}^{N} p(j) x(j),
\]
subject to
\[
\sum_{j=1}^{N} w(i,j) x(j) \leq c(i), \quad \text{for all } i \in [D].
\]

\[
x(j) \in \{0,1\}, \quad \text{for all } j \in [N].
\]

Here, \((c(i))_{i=1}^{D}\) is the list of knapsack capacities, \((w(i,j))_{j=1}^{N} : i \in [D]\) is the list of corresponding weights and \((x(j))_{j=1}^{N}\) is the list of binary valued decision variables. In addition, the weight coefficients \((w(i,j))_{j=1}^{N} : i \in [D]\), as well as the knapsack capacities \((c(i))_{i=1}^{D}\) are all positive integers. In the following \(z^*, x^*\) denote the optimal objective function value and solution respectively. The parameters \((p(j))_{j=1}^{N} \subseteq (0, \infty)^N\) are known as the profits. Finally, in the sequel one of the d-KP instances will be denoted by \(\Pi = \langle(c(i))_{i=1}^{D}, (p(j))_{j=1}^{N}, \{w(i,j) : j \in [N], i \in [D]\}\rangle\).

The following efficiency coefficients are introduced, scaled according to the respective constraint capacities
Definition 1 (d-KP efficiency coefficients). For any instance of d-KP (Problem 1), the efficiency coefficients are defined by
\[ g(j) \overset{\text{def}}{=} \sum_{i=1}^{D} \frac{p(j)}{w(i,j)} c(i), \quad j \in [N]. \] (5)

Before continuing the analysis, the next hypothesis is adopted.

Hypothesis 1. In the sequel it is assumed that the instances \( \Pi \) of the d-KP satisfy the following conditions
\[ w(i,j) \leq c(i), \quad \text{for all } i \in [N], j \in [D], \] (6a)
\[ \sum_{j=1}^{N} w(i,j) > c(i), \quad \text{for all } i \in [D]. \] (6b)

Remark 1 (d-KP Setting). The condition (6) in Hypothesis 1 guarantees two things. First, every item is eligible to be chosen (Inequality (6a)). Second, none of the constraints can be dropped (Inequality (6b)).

2.1. The Divide-and-Conquer Approach

The Divide-and-Conquer method for solving the 0-1KP was introduced in [1]. There, an extensive discussion (theoretical and empirical) was presented on the possible strategies to implement it. Here, its conclusions are merely adjusted to yield the following method.

Definition 2 (Divide-and-Conquer Method for d-KP). Let \( \Pi = \langle (c(i))_{i \in [D]}, (p(j))_{j \in [N]}, \{w(i,j) : j \in [N], i \in [D]\} \rangle \) be an instance of Problem 1
(i) Sort the items in decreasing order according to the efficiency coefficients and re-index them so that
\[ g(1) \geq g(2) \geq \ldots \geq g(N). \] (7)
(ii) Define \( V_{\text{left}} = \{ j \in [N] : j \text{ is odd} \} \), \( V_{\text{right}} = \{ j \in [N] : j \text{ is even} \} \) and
\[ c_{\text{left}}(i) = \left[ \frac{\sum_{j \in V_{\text{left}}} w(i,j)}{\sum_{j=1}^{N} w(i,j)} \right], \quad c_{\text{right}}(i) = c(i) - c_{\text{left}}(i), \quad \text{for all } i \in [D]. \]
(iii) A Divide-and-Conquer pair of Problem 1 is the couple of subproblems \( \Pi_{\text{side}} : \text{side} \in \{ \text{left}, \text{right} \} \), each with input data \( \Pi_{\text{side}} = \langle (c_{\text{side}}(j))_{j \in [D]}, (p(j))_{j \in [N]}, \{w(i,j) : j \in V_{\text{side}} \} \rangle \). In the sequel, \( (\Pi_{\text{side}}, s = \text{left}, \text{right}) \) is referred as a D&C pair and denote by \( z_{\text{side}}^* \) the optimal solution value of the problem \( \Pi_{\text{side}} \).
(iv) The D&C solution is given by
\[ x_{\text{DC}}^{\text{mt}} \overset{\text{def}}{=} x_{\text{left}}^{\text{mt}} \cup x_{\text{right}}^{\text{mt}}, \quad z_{\text{DC}}^{\text{mt}} \overset{\text{def}}{=} z_{\text{left}}^{\text{mt}} + z_{\text{right}}^{\text{mt}}. \] (8)

Here, the index mt indicates the method used to solve the problem, in this work and particular problem, only \( \text{mt} = * \) is used. Also, some abuse of notation is introduced, denoting by \( x_{\text{side}}^{\text{mt}} \), the solution of \( \Pi_{\text{side}} \) and using the same symbol as a set of chosen items (instead of a vector) in the union operator. In particular, the maximal possible value occurs when all the summands are at its max i.e., the method approximates the optimal solution by the feasible solution \( x_{\text{DC}}^{\text{mt}} = x_{\text{left}}^{\text{mt}} \cup x_{\text{right}}^{\text{mt}} \), with objective value \( z_{\text{DC}}^{\text{mt}} = z_{\text{left}}^{\text{mt}} + z_{\text{right}}^{\text{mt}} \).
Table 1: d-KP problem of Example 1

|   | 1   | 2 | 3 | 4 | 5 | 6 |
|---|-----|---|---|---|---|---|
| $\rho(j)$ | 5   | 11| 11| 71| 2 | 2 |
| $w(1,j)$ | 6   | 7 | 7 | 7 | 4 |   |
| $w(2,j)$ | 4   | 1 | 1 | 6 | 1 | 8 |
| $g(j)$   | 13.53 | 41.01 | 144.03 | 14.28 | 7.46 | 4.14 |

Example 1 (Divide-and-Conquer Method on d-KP). Consider the d-KP instance described by the table with knapsack capacities $c(1) = 16$, $c(2) = 11$ and number of items $N = 6$. In this particular case the D&C pair is given by

$\Pi_{\text{left}} : \quad \mathcal{V}_{\text{left}} = [3, 4, 5], \quad \alpha_{\text{left}}(1) = 8, \quad \alpha_{\text{left}}(2) = 5,$

$\Pi_{\text{right}} : \quad \mathcal{V}_{\text{right}} = [2, 1, 6], \quad \alpha_{\text{right}}(1) = 8, \quad \alpha_{\text{right}}(2) = 6.$

Proposition 3. Let $\Pi = \langle (c(i))_{i \in [D]}, (\rho(j))_{j \in [N]}, \{w(i,j) : j \in [N], i \in [D]\} \rangle$ be an instance of d-KP (problem 1). Let $(\Pi_{\text{left}}, \Pi_{\text{right}})$ be the associated Divide-and-Conquer pair introduced in Definition 2. Then,

$$x_{\text{DC}}^{mt} = x_{\text{left}}^{mt} \cup x_{\text{right}}^{mt} \text{ is } \Pi \text{-feasible,}$$

for $mt = G, *, \text{LR}$, \hspace{1cm} (9a)

$$z_{\text{left}}^{mt} + z_{\text{right}}^{mt} = z_{\text{DC}}^{mt} \leq z^{mt},$$

for $mt = *, \text{LR}$. \hspace{1cm} (9b)

Here, G and LR indicate any greedy method and LR the linear relaxation.

Proof. Trivial. \hspace{1cm} $\Box$

As already stated, the purpose of this work is to determine experimentally, under which conditions an instance $\Pi$ of d-KP can be solved using the Divide-and-Conquer method in an acceptable way. That is, if the trade between computational complexity and accuracy is acceptable. To this end, a set of parameters is introduced, which has proved to be useful in order to classify the d-KP instances, within these classifications the output of numerical methods is comparable, see [30].

Definition 3 (Tightness Ratios). Let $\Pi = \langle (c(i))_{i \in [D]}, (\rho(j))_{j \in [N]}, \{w(i,j) : j \in [N], i \in [D]\} \rangle$ be an instance of d-KP, then the set of tightness ratios is given by

$$t(i) \overset{\text{def}}{=} \frac{c(i)}{\sum_{j=1}^{N} w(i,j)},$$

for all $i \in [D]$. \hspace{1cm} (10)

(Observable the differences with respect to the efficiency coefficients introduced in Definition 1.)

2.2. Numerical Experiments

In order to numerically asses the Divide-and-Conquer method’s effectiveness on d-KP, the numerical experiments are designed according to its main parameters i.e., size ($N \in \{6, 10, 20, 50, 100, 250, 500, 1000\}$), restrictiveness ($D \in \{2, 3, 4, 5, 6\}$) and tightness ($t = \{0.25, 0.5, 0.75\}$). While the values of the first three parameters $N, D, t$ were decided based on experience (see [30]), the number of Bernoulli trials $k = 1000$ was decided based on Equation (2). The variance $\sigma^2$ was chosen as the worst possible value for a sample of 100 trials, which delivered an approximate value $k = 1000$. (After, executing the 1000 experiments the variance showed consistency with the initial value computed from the 100 trial-sample.)

The random instances $\Pi_\ell = \langle (c(i))_{i \in [D]}, (\rho(j))_{j \in [N]}, \{w(i,j) : j \in [N], i \in [D]\} \rangle$ were generated as follows.

a. The profits $(P(i))_{i \in [N]}$ will be independent random variables with uniform distribution on the interval $[1, N \cdot D]$ (i.e., $P(P(i) = \ell) = \frac{1}{N \cdot D}$ for all $\ell \in [1, N \cdot D]$ and $i \in [N]$).
b. The capacities \((C_i)_{i \in [D]}\) are independent random variables uniformly distributed on the interval \([1, N \cdot D]\).

c. For all \(i \in [D]\), the weights \((W(i,j))_{i \in [M]}\) are almost independent random variables, uniformly distributed in the interval \([1, C(i)]\) and such that \(t(i) \approx C(i) / \sum_{j=1}^{N} W(i,j)\), where \(t(i)\) is the tightness ratio introduced in Definition 3.

**Definition 4 (d-KP Performance Coefficients).** The d-KP performance coefficients are given by the percentage fractions of the numerical solution and the computational time given by the Divide-and-Conquer method with respect to the exact solution when using the same method of resolution for both cases, i.e.,

\[
S_f = 100 \times \frac{z_{DC}}{z^*}, \quad T_f = 100 \times \frac{T_{DC}}{T^*}.
\]

The tables 2, 3, 4 below, summarize the expected values of the performance coefficients \(S_f\) (solution fraction) and \(T_f\) (time fraction) (see Definition 4) for the tightness values \(t = 0.25, 0.5, 0.75\) respectively and dimensions \(D = 2, 4, 6\). The corresponding graphs are depicted in the figures 1, 2 and 3.

| Items | \(D = 2\) | \(D = 4\) | \(D = 6\) |
|-------|-------|-------|-------|
| \(N\) | \(S_f\) | \(T_f\) | \(S_f\) | \(T_f\) | \(S_f\) | \(T_f\) |
| 6     | 51.85 | 63.25 | 8.39  | 46.87 | 33.18 | 51.28 |
| 10    | 76.39 | 54.63 | 60.88 | 54.42 | 70.75 | 48.34 |
| 20    | 88.98 | 54.13 | 82.84 | 51.43 | 87    | 53.65 |
| 50    | 96.36 | 51.34 | 93.69 | 52.39 | 95.69 | 51.96 |
| 100   | 98.67 | 52.09 | 97.2  | 51.49 | 97.53 | 51.84 |
| 250   | 99.63 | 53.47 | 99.03 | 51.65 | 99.01 | 51.86 |
| 500   | 99.84 | 51.63 | 99.55 | 50.9  | 99.47 | 51.07 |
| 1000  | 99.94 | 51.41 | 99.81 | 50.23 | 99.77 | 50.52 |

Table 2: Performance table for the Divide-and-Conquer method, solving d-KP problem. Tightness ratio \(t = 0.25\). Bernoulli trials \(k = 1000\). The performance coefficients \(S_f\) and \(T_f\) are introduced in Equation (11).

![Figure 1: Graphic solution fraction \(S_f\) and time fraction \(T_f\) for the values displayed in table 2. The domain is given by the number of items \(N\) in logarithmic scale.](image)

3. The Bin Packing Problem (BPP)

In the current section the Bin Packing Problem (BPP) is introduced and a list of greedy algorithms is reviewed, which will be used in the following to compute/measure the expected performance of the Divide-and-Conquer method acting on BPP.

The Bin-Packing Problem is described as follows. Given \(N\) objects, each of a given weight \((w(j))_{j=1}^{N} \subseteq (0, c]\) and bins of capacity \(c\) (at least \(N\) of them), the aim is to assign the objects in a way that the minimum number of bins is used and that every object is packed in one of the bins. Without loss of generality, the capacity of the items can be normalized to \(c = 1\) and the weight of the items accordingly. Hence, the BPP formulates as follows

**Problem 2 (Bin Packing Problem, BPP).** Consider the problem

\[
z^* \overset{\text{def}}{=} \min \sum_{i=1}^{N} y(i),
\]

(12a)
Table 3: Performance table for the Divide-and-Conquer method, solving d-KP problem. Tightness ratio $t = 0.5$. Bernoulli trials $k = 1000$. The performance coefficients $S_f$ and $T_f$ are introduced in Equation (11).

Table 4: Performance table for the Divide-and-Conquer method, solving d-KP problem. Tightness ratio $t = 0.75$. Bernoulli trials $k = 1000$. The performance coefficients $S_f$ and $T_f$ are introduced in Equation (11).

subject to

$$\sum_{j=1}^{N} w(j) x(i,j) \leq y(i),$$

for all $i \in [N]$. \hspace{1cm} (12b)

$$x(i,j), y(i) \in \{0, 1\},$$

for all $i, j \in [N]$. \hspace{1cm} (12c)

Here, $(w(j))_{j=1}^{N} \subseteq (0, 1]$ is the list of item weights, $\{x(i,j) : i,j \in [N]\}$ is the list of binary valued item assignment decision variables, with $x(i,j) = 1$ if item $j$ is assigned to bin $i$ and $x(i,j) = 0$ otherwise. Finally, $(y(i))_{i=1}^{N}$ is the list of binary valued bin-choice decision variables, setting $y(i) = 1$ if bin $i$ is nonempty and $y(i) = 0$ otherwise.
As it is well-known, most of the literature is concerned with heuristic methods for solving BPP (see [16]). Therefore, in this work is focused on the interaction of the Divide-and-Conquer method with the three most basic heuristic algorithms: Next Fit Decreasing (NFD), First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) (see [31] and [32]). For the sake of completeness these algorithms are described below. In the three cases, it is assumed that the items are sorted decreasingly according their weights, i.e. \( w(1) \geq w(2) \geq \ldots \geq w(N) \).

a. **Next Fit Algorithm.** The first item is assigned to bin 1. Each of the items 2, \ldots, N is handled as follows: the item \( j \) is assigned to the current bin if it fits; if not it is assigned to a new bin (next fit).

b. **First Fit Algorithm.** Consider the items increasingly according to its index. For each item \( j \), assign it to the first (first fit) initialized bin \( i \) where it fits (if any), and if does not fit in any initialized bin, assign it to a new one.

c. **Best Fit Algorithm.** Consider the items increasingly according to its index. For each item \( j \), assign it to the bin \( i \) where it fits (if any) and it is as full as possible, and if it does not fit in any initialized bin, assign it to a new one.

3.1. The Divide-and-Conquer Approach

The NF, FF and BF algorithms work exactly as NFD, FFD and BFD, except that they do not act on a sorted list of items. It has been established that the NF algorithm (and consequently FF as well as BF) has good performance on items of small weight (see [32]). Therefore, it is only natural to deal with the pieces according to their weight in decreasing order i.e., in the BPP problem the weights are used as greedy function (or efficiency coefficients) in order to construct the corresponding Divide-and-Conquer method.

**Definition 5 (Divide-and-Conquer Method for BPP).** Let \( \Pi = \langle (w(j))_{j \in [N]} \rangle \) be an instance of Problem 2

(i) Sort the items in decreasing order according to their weights and re-index them so that \( w(1) \geq w(2) \geq \ldots \geq w(N) \).

(ii) Define \( V_{\text{left}} = \{ j \in [N] : j \text{ is odd} \} \) and \( V_{\text{right}} = \{ j \in [N] : j \text{ is even} \} \).

(iii) A Divide-and-Conquer pair of Problem 1 is the couple of subproblems \( (\Pi_{\text{side}} : \text{side} \in \{ \text{left}, \text{right} \}) \), each with input data \( \Pi_{\text{side}} = \langle (w(j))_{j \in V_{\text{side}}} \rangle \). In the sequel, \( (\Pi_{\text{side}}, s = \text{left, right}) \) is referred as a D&C pair and denote by \( z_{\text{alg}}^{\text{left}} \) the solution value of the problem \( \Pi_{\text{side}} \), while \( z_{\text{alg}} \) denotes the solution value of the full problem \( \Pi \), furnished by the algorithm alg.

(iv) The D&C solution of Problem 2 is given by

\[
\begin{align*}
    x_{\text{DC}}^{\text{alg}} & \overset{\text{def}}{=} x_{\text{left}}^{\text{alg}} \cup x_{\text{right}}^{\text{alg}}, \\
z_{\text{DC}}^{\text{alg}} & \overset{\text{def}}{=} z_{\text{left}}^{\text{alg}} + z_{\text{right}}^{\text{alg}}.
\end{align*}
\]

Here, the index mt indicates the method used to solve the problem. This work, uses \( \text{alg} = \text{NFD, FFD, BFD} \), standing for the Next Fit Decreasing, First Fit Decreasing and Best Fit Decreasing algorithms respectively (described in the previous section). Also, some abuse of notation is introduced, denoting by \( x_{\text{side}}^{\text{alg}} = \{ x_{\text{side}}^{\text{alg}}(i,j) : i,j \in V_{\text{side}} \} \) the optimal solution of \( \Pi_{\text{side}} \) and using the same symbol as a set of chosen items (instead of a vector) in the union operator. In particular, the minimal possible value occurs when all the summands are at its minimum i.e., the method approximates the optimal solution by the feasible solution \( x_{\text{DC}}^{\text{alg}} = x_{\text{left}}^{\text{alg}} \cup x_{\text{right}}^{\text{alg}} \) with objective value \( z_{\text{DC}}^{\text{alg}} = z_{\text{left}}^{\text{alg}} + z_{\text{right}}^{\text{alg}} \).

**Example 2 (Divide-and-Conquer Method on BPP).** Consider the BPP instance described by the table 5, number of items \( N = 6 \).

For this particular case the D&C pair is given by

\[
\begin{align*}
    \Pi_{\text{left}} : & \quad V_{\text{left}} = [5, 1, 3], \\
    \Pi_{\text{right}} : & \quad V_{\text{right}} = [2, 6, 4].
\end{align*}
\]
Table 5: BPP problem of Example 2, number of items $N = 6$.

| $j$ | $w(j)$ |
|-----|--------|
| 1   | 0.5    |
| 2   | 0.7    |
| 3   | 0.25   |
| 4   | 0.1    |
| 5   | 0.85   |
| 6   | 0.31   |

Proposition 4. Let $\Pi = \langle (w(j))_{j \in [N]} \rangle$ be an instance of the BPP problem. Let $(\Pi_{\text{left}}, \Pi_{\text{right}})$ be the associated Divide-and-Conquer pair introduced in Definition 5. Then,

$$x_{\text{alg}}^{\text{DC}} = x_{\text{alg}}^{\text{left}} \cup x_{\text{alg}}^{\text{right}} \text{ is } \Pi\text{-feasible},$$

for alg = NFD, FFD, BFD. (15)

Proof. Trivial. \qed

Remark 2. Observe that, unlike Proposition 3, here, there is no claim of an inequality of control such as (9b). This is because the NFD, FFD and BFD methods are all heuristic. Hence, it can happen (and it actually does for some few instances) that $z_{\text{alg}}^{\text{DC}} \geq z_{\text{alg}}$ for alg = NFD, FFD, BFD; it is worth noticing that this phenomenon is more frequent for the NFD method.

3.2. Numerical Experiments

In order to numerically assess the Divide-and-Conquer method’s effectiveness on BPP, the numerical experiments are designed according to its main parameter i.e., size ($N \in \{20, 50, 100, 250, 500, 1000, 1500, 2000\}$) and number of Bernoulli trials ($k$). The number of items was decided based on experience (more specifically the Divide-and-Conquer method yields poor results for low values of $N$ as it can be seen in the results), while the number of Bernoulli trials $k = 1500$ was chosen based on Equation (2). The variance $\sigma^2$ was chosen as the worst possible value for a sample of 100 trials, which delivered an approximate value $k = 1500$. (After, executing the 1500 experiments the variance showed consistency with the the initial value computed from the 100 trial-sample.)

For the random instances $\Pi = \langle (W(j))_{j \in [N]} \rangle$, the weights $(W(j))_{j \in [N]}$ are independent random variables uniformly distributed on the interval $[0, 1]$ (notice that $\mathbb{P}(W(i) = 0) = 0$ for all $i \in [N]$). Next, the numerical and time performance coefficients are introduced.

Definition 6 (BPP Performance Coefficients). The BPP performance coefficients are given by the percentage fractions of the solution and the computational time given by the Divide-and-Conquer method with respect to the exact solution when using the same method of resolution for both cases, i.e.,

$$S^\text{alg}_f = 100 \times \frac{z^\text{alg}}{z^\text{DC}}, \quad T^\text{alg}_f = 100 \times \frac{T^\text{alg}_\text{DC}}{T^\text{alg}},$$

for alg = NFD, FFD, BFD. (16)

Here, $z^\text{alg}, z^\text{DC}$ indicate the solution of furnished for the full problem and for the Divide-and-Conquer method respectively, when using the algorithm alg.

Remark 3. The definition of $S^\text{alg}_f$ (equation (16)) for BPP defers from that given in d-KP (equation (11)) in order to keep normalized the fraction. Since BPP is a minimization problem the Divide-and-Conquer solution will be larger than the exact solution ($z^\text{DC} \geq z^\text{alg}$) for most of the cases. This contrasts with d-KP which is maximization problem, where the opposite behavior takes place ($z^\text{DC} \leq z^\ast$).

The table below, summarizes the expected values of the performance coefficients $S^\text{alg}_f$ (solution fraction) $T^\text{alg}_f$ (time fraction) (see Definition 6). The corresponding graphs are depicted in the figures 4(a) for $S^\text{alg}_f$ and (b) for $T^\text{alg}_f$, respectively.
Table 6: Performance table for the Divide-and-Conquer method, when solving the BPP problem with the three methods: NFD, FFD and BFD. Bernoulli trials \( k = 1500 \). The performance coefficients \( S_{alg}^f \) and \( T_{alg}^f \) for \( alg = NFD, FFD, BFD \) are introduced in Equation (16) by the authors. Note that the values are rounded to two decimal places.

| Items | N | NFD | FFD | BFD |
|-------|---|-----|-----|-----|
| \( N \) | \( S_{NFD}^f \) | \( T_{NFD}^f \) | \( S_{FFD}^f \) | \( T_{FFD}^f \) | \( S_{BFD}^f \) | \( T_{BFD}^f \) |
| 20    | 99.03 | 126.86 | 96.57 | 70.56 | 96.57 | 92.7 |
| 50    | 99.77 | 112.98 | 98.55 | 59.75 | 98.59 | 82.73 |
| 100   | 99.83 | 111.01 | 98.92 | 54.66 | 98.92 | 76.72 |
| 250   | 99.99 | 102.36 | 99.59 | 52.09 | 99.58 | 65.09 |
| 500   | 99.99 | 99.97 | 98.84 | 50.26 | 98.84 | 58.35 |
| 1000  | 99.98 | 100.04 | 99.92 | 50.43 | 99.92 | 55.04 |
| 1500  | 99.99 | 101.05 | 99.94 | 50.83 | 99.94 | 53.85 |
| 2000  | 99.99 | 101.16 | 99.96 | 50.76 | 99.96 | 53.05 |

Figure 4: The graphs above depict the approximate expected performance for \( k = 1500 \) Bernoulli trials. In figure (a) the solution fraction \( S_{alg}^f \), in figure (b) the time fraction \( T_{alg}^f \) coefficients are displayed, for the BPP problem (see Definition 6) and \( alg = NFD, FFD, BFD \). The domain \( N \) is given in logarithmic scale, the values are extracted from the table.

4. The Traveling Salesman Problem TSP

The last problem to be analyzed under the Divide-and-Conquer method is the Traveling Salesman Problem TSP. Some graph-theory definitions are recalled in order to describe it.

**Definition 7.**

(i) An edge-weighted, directed graph is a triple \( G = (V, E, d) \), where \( V \) is the set of vertices, \( E \subseteq V \times V \) is the set of (directed) edges or arcs and \( d : E \rightarrow \mathbb{R}^+ \) is a distance function assigning each arc \( e \in E \) a distance \( d(e) \).

(ii) A path is a list \( (u_1, ..., u_k) \) of vertices \( u_i \in V \) for all \( i = 1, ..., k \), holding \( (u_i, u_{i+1}) \in E \) for \( i = 1, ..., k - 1 \).

(iii) A Hamiltonian cycle in \( G \) is a path \( p = (u_1, ..., u_k, u_1) \) in \( G \), where \( k = |V| \) and \( \bigcup_{i=1}^{k} u_i = V \) (i.e., each vertex is visited exactly once except for \( u_1 \)).

(iv) Given an edge-weighted directed graph \( G = (V, E, d) \) and a vertex subset \( U \) of \( V \), the induced subgraph, denoted by \( G(U) = (U, E_U, d_U) \) is the subgraph of \( G \) whose vertex set is \( U \), whose edge set consists on
Problem 3 (The Traveling Salesman Problem, TSP). Given a directed graph $G = (V, E, d)$ with $n = |V|$ vertices, the TSP is to find a Hamiltonian cycle $(u_1, \ldots, u_n, u_1)$ such that the cost $C(u_1, \ldots, u_n, u_1) = d((u_k, u_l)) + \sum_{i=1}^{n-1} d((u_i, u_{i+1}))$ (the total sum of the weights of the edges in the cycle) is minimized.

(i) The TSP is said to be symmetric if $d((u, v)) = d(v, u)$ for all $u, v \in V$. Otherwise, it is said to be asymmetric.

(ii) The TSP is said to be metric if $d((u, v)) \leq d((u, w)) + d((w, v))$ for all $u, v, w \in V$.

In the current work three cases are analyzed:

a. MS, when the TSP is metric and symmetric.

b. MA, when the TSP is metric but not symmetric.

c. NMS, when the TSP is not metric but it is symmetric.

Since it is so widespread and contributes little to our discussion, the formulation of TSP is omitted (see Section 10.3 in [33] for a formulation).

4.1. The Divide-and-Conquer Approach

In order to apply the Divide-and-Conquer approach on the TSP it is necessary to introduce an efficiency coefficient to break the original problem in two sub-problems. Hence, in the TSP case, the efficiency for each vertex of the graph, is defined as the sum of the distances of all the edges incident on the node.

Definition 8 (TSP efficiency coefficients). Given an edge-weighted directed graph $G = (E, V, d)$, for each node $u \in V$ its efficiency coefficient is defined as

$$g(u) = \sum_{v \in V - \{u\}} d((u, v)) + d((v, u)).$$

(17)

Next, a greedy algorithm needs to be introduced in order merge two cycles (the two solution cycles corresponding to each sub-problem: $\Pi_{\text{left}}, \Pi_{\text{right}}$).

Definition 9 (Cycle Greedy Merging). Let $G = (V, E, d)$ be a graph with $|V| = N$ and $|E| = K$. Let $V_{\text{left}}, V_{\text{right}}$ be a subset partition of $V$ with $|V_{\text{left}}| = p$, $|V_{\text{right}}| = q$ and let

$$C_{\text{left}} = (u_1, \ldots, u_p, u_1), \quad C_{\text{right}} = (v_1, \ldots, v_q, v_1),$$

(18)

be two cycles contained in $V_{\text{left}}$ and $V_{\text{right}}$ respectively. Let $(u_i, u_{i+1}), (v_j, v_{j+1})$ be the most expensive arcs in $C_{\text{left}}$ and $C_{\text{right}}$ respectively and break possible ties choosing the lowest index. (Here, the indexes $i, i + 1$ and $j, j + 1$ are understood in mod $p$ and mod $q$ respectively.) Define the merged cycle as

$$C = (u_1, \ldots, u_{i-1}, u_i, v_{j+1}, v_{j+2}, \ldots, v_q, v_1, \ldots, v_{j-1}, v_j, u_{i+1}, u_{i+2}, \ldots, u_p).$$

(19)

Where the arcs $(u_i, v_{j+1}), (v_j, u_{i+1})$ (marked with underbraces) were included to replace the corresponding arcs $(u_i, u_{i+1})$ and $(v_j, v_{j+1})$. (See Figure 5 for an example.)

Remark 4. It must be observed that the merging process is introduced in Definition [9] well-defined, because the choice of arcs to be removed is unique as well as the inclusion of the new arcs, marked with underbraces in [19]. Once the arcs $(u_i, u_{i+1}), (v_j, v_{j+1})$ are removed, the pair of arcs $(u_i, v_{j+1}), (v_j, u_{i+1})$ are the only possible choice for respective replacement that would deliver a joint cycle, given the orientations of $C_{\text{left}}$ and $C_{\text{right}}$. This observation is particularly important for the asymmetric instances of TSP (MA).
Definition 10 (Divide-and-Conquer Method for TSP). Let $\Pi = (V,E,d)$ be an instance of Problem 3

(i) Sort the vertices in increasing order according to their efficiencies, that is

$$g(v_{\pi(1)}) \leq g(v_{\pi(2)}) \leq \ldots \leq g(v_{\pi(N)}),$$

where $\pi \in S_N$ is an adequate permutation and $V = \{v_i : i \in [N]\} = \{v_{\pi(i)} : i \in [N]\}$.

(ii) Define $V_{\text{left}} = \{v_{\pi(i)} : i \text{ is odd}\}$ and $V_{\text{right}} = \{v_{\pi(i)} : i \text{ is even}\}$.

(iii) A Divide-and-Conquer pair of Problem 3 is the couple of subproblems $(\Pi_{\text{side}} : \text{side} \in \{\text{left, right}\})$, each with input data $\Pi_{\text{side}} = G(V_{\text{side}})$, i.e., the induced subgraph introduced in Definition 7. In the following, $(\Pi_{\text{side}}, s = \text{left, right})$ is referred as a D&C pair and denote by $z_s$ the optimal solution value of the problem $\Pi_{\text{side}}$.

(iv) The D&C solution of Problem 3 is given by the greedy merging of cycles $C_{\text{left}}, C_{\text{right}}$ (see Definition 9) furnished by the solution of $\Pi_{\text{left}}$ and $\Pi_{\text{right}}$ respectively. (See Example 3 and Figure 5 for an illustration.)

Example 3 (Divide-and-Conquer Method on TSP). Consider the TSP instance described by the table 7 with $N = 6$ vertices. For this particular case the following order holds: $g(v_3) \leq g(v_5) \leq g(v_2) \leq g(v_6) \leq g(v_1) \leq g(v_4)$. Therefore,

| $v_i$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ |
|-------|-------|-------|-------|-------|-------|-------|
| $v_1$ | 0.00  | 0.61  | 1.08  | 0.46  | 0.11  |       |
| $v_2$ | 0.61  | 0.00  | 0.53  | 0.71  | 0.17  | 0.54  |
| $v_3$ | 0.10  | 0.53  | 0.00  | 0.98  | 0.39  | 0.12  |
| $v_4$ | 1.08  | 0.71  | 0.98  | 0.00  | 0.83  | 1.07  |
| $v_5$ | 0.46  | 0.17  | 0.39  | 0.83  | 0.00  | 0.38  |
| $v_6$ | 0.11  | 0.54  | 0.12  | 1.07  | 0.38  | 0.00  |

Table 7: TSP problem of Example 3 number of items $N = 6$. Distance matrix table.

$\Pi_{\text{left}} : V_{\text{left}} = \{v_3, v_2, v_1\}$.

$\Pi_{\text{right}} : V_{\text{right}} = \{v_5, v_6, v_4\}$.

Given that this is a symmetric instance (actually MS) of TSP and that $\Pi_{\text{left}}$ and $\Pi_{\text{right}}$ have both three vertices, there is only one possible solution for each TSP; namely $C_{\text{left}} = (v_1, v_2, v_3)$ and $C_{\text{right}} = (v_4, v_6, v_5)$ respectively. See Figure 5. Next, observe that the most expensive arc in $C_{\text{left}}$ is $(v_1, v_2)$, while the heaviest arc in $C_{\text{right}}$ is $(v_4, v_5)$. Hence, these two arcs need to be removed (depicted in dashed line in Figure 5) and replaced by $(v_1, v_5), (v_4, v_2)$ (depicted in blue in Figure 5) respectively. Observe that this is the only possible choice in order to preserve the direction of the cycles, in particular the remaining arcs (depicted in red in Figure 5) would fail to build a global Hamiltonian cycle.

4.2. Numerical Experiments

In order to numerically assess the Divide-and-Conquer method’s effectiveness on TSP, the numerical experiments are designed according to its main parameters i.e., size ($N \in \{8, 18, 30, 44, 60, 78, 98, 120\}$) and number of Bernoulli trials ($k$). The number of items was decided based on experience and observation of the method’s performance; while the number of Bernoulli trials $k = 2500$ was chosen based on Equation (2). The variance $\sigma^2$ was chosen as the worst possible value for a sample of 150 trials, which delivered an approximate value $k = 2500$. (After, executing the 2500 experiments the variance showed consistency with the initial value computed from the 150 trial-sample.)

The random instances were be generated as follows
Figure 5: Merging Cycles Schematics for Example 3 (see Fig. 2 for a general definition). The cycles $C_{\text{left}} = (v_1, v_2, v_3)$ and $C_{\text{right}} = (v_4, v_5, v_6)$ are to be merged. The most expensive arcs for each, depicted in dashed line are removed and replaced by the only possible pair, in order to build a global Hamiltonian Cycle. Choosing other pair of arcs (depicted in red) would fail to assemble a cycle due to the orientation.

a. For the MS instances of TSP, $N$ points in the square $[0, 1] \times [0, 1] \subseteq \mathbb{R}^2$ are generated (using the uniform distribution) and compute the distance matrix $D_{\text{MS}}$.

b. For the MA instances of TSP, $N$ points on the unit circle $S^1 = \{\vec{x} : ||\vec{x}|| = 1\} \subseteq \mathbb{R}^2$ are generated (using the uniform distribution) and compute the asymmetric distance matrix $D_{\text{MA}}$. Given $\vec{x}, \vec{y} \in S^1$, the asymmetric distance $d_{\text{MA}}(\vec{x}, \vec{y})$ is defined as the length of the shortest clockwise path from $\vec{x}$ to $\vec{y}$ contained in $S^1$. Clearly $d_{\text{MA}}(\vec{x}, \vec{y}) = d_{\text{MA}}(\vec{y}, \vec{x})$ unless $\vec{x}$ and $\vec{y}$ are antipodal points.

c. For the NMS instances of TSP, an $N \times N$ upper triangular matrix $M$ is generated, whose entries are uniformly distributed in $[0, 1]$ and its diagonal is null. The distance matrix is defined as $D_{\text{NMS}} = M + M^T$.

The performance coefficients are analogous to those introduced in Definition 6, given that TSP is a minimization problem as BPP (see also Remark 3).

**Definition 11 (TSP Performance Coefficients).** The TSP performance coefficients are given by the percentage fractions of the solution and the computational time given by the Divide-and-Conquer method with respect to the exact solution when using the same method of resolution for both cases, i.e.,

$$S^\text{case}_f = 100 \times \frac{z^\text{case}}{z^\text{DC}}, \quad T^\text{case}_f = 100 \times \frac{T^\text{case}}{T^\text{DC}},$$

where $S^\text{case}_f$ and $T^\text{case}_f$ indicate the solution of furnished for the full problem and for the Divide-and-Conquer method respectively, when solving the respective case.

The table below summarizes the expected values of the performance coefficients $S^\text{case}_f$ (solution fraction) $T^\text{case}_f$ (time fraction). The corresponding graphs are depicted in the figures (a) for $S^\text{case}_f$ and (b) for $T^\text{case}_f$, respectively.

5. Conclusions and Final Discussion

The present work yields several conclusions which are listed below

(i) For d-KP, the tables 2, 3, 4, as well as their corresponding figures 1, 2, 3 show that the Divide-and-Conquer method is recommendable, starting from certain number of items $N_0$. The latter, according to the required precision when solving the problem. It is also clear that the tightness plays a very important role in defining the number $N_0$, starting from which the Divide-and-Conquer method is useful. Although the dimension $D$ also has an impact on $N_0$, it is clear that for different values of $D$, the expected values of the method’s performance get close to each other for $N \geq 100$, which is reasonably low.
Table 8: Performance table for the Divide-and-Conquer method, solving the TSP problem for the three types of instances: MS, MA and NMS. Bernoulli trials \( k = 2500 \). The performance coefficients \( S^\text{case} \) and \( T^\text{case} \) for case = MS, MA, NMS are introduced in Equation (21).

| Items | MS | MA | NMS |
|-------|----|----|-----|
| \( N \) | \( S^\text{MS} \) | \( T^\text{MS} \) | \( S^\text{MA} \) | \( T^\text{MA} \) | \( S^\text{NMS} \) | \( T^\text{NMS} \) |
| 8     | 70.15 | 75.66 | 64  | 106.98 | 59.62 | 86.32 |
| 18    | 65.72 | 79.64 | 61.37 | 69.37 | 60.59 | 105.08 |
| 30    | 68.2  | 64.61 | 87.9  | 53.11 | 67.05 | 92.1  |
| 44    | 70.26 | 67.7  | 84.12 | 95.54 | 70.74 | 80.52 |
| 60    | 70.04 | 62.68 | 98.85 | 84.4  | 74.82 | 75.35 |
| 78    | 69.69 | 55.43 | 94.45 | 66.68 | 77.22 | 63.11 |
| 98    | 70.24 | 56.6  | 97.86 | 68.91 | 78.65 | 70.58 |
| 120   | 71.52 | 50.57 | 99.94 | 77.16 | 78.44 | 57.31 |

Figure 6: The graphs above depict the approximate expected performance for \( k = 2500 \) Bernoulli trials. In figure (a) the solution fraction \( S^\text{case} \), in figure (b) for the time fraction \( T^\text{case} \) coefficients are displayed, for the three types of instance of TSP case = MS, MA, NMS. Unlike the previous examples, the domain \( N \) is given in linear scale. The values are extracted from the table 8.

(ii) In the case of the BPP, it is clear that Divide-and-Conquer has a poor interaction with the NFD algorithm, because the required computational time is always above 100% and a precision price is always paid. On the other hand, the FFD algorithm complements very well the Divide-and-Conquer method, giving substantial reduction of computational time for a low number of items, namely \( N \geq 50 \). Finally, the BFD algorithm is a bad complement for the Divide-and-Conquer method, compared with FFD algorithm. The latter, considering that BFD yields a marginal improvement of precision with respect to FFD, while using a significant higher fraction of computational time (although not as much as NFD). It is to be noted that the three algorithms yield significantly high precision values, even for low values of \( N \) and that ranking their usefulness is decided based on its computational time. All of the previous can be observed in the table and the figure.

(iii) Analyzing the results for TSP, reported in table and figure, it is immediate to conclude that the Divide-and-Conquer method performs poorly for the MA case. It shows an erratic curve of computational time, although it attains a good precision levels for \( N \geq 50 \) vertices. As for the other two cases, although...
they show reasonable computational time fraction for $N \geq 80$, the precision fraction never goes above 80%, which is very low value to consider Divide-and-Conquer as a valid option for solving TSP. Therefore, it follow that the method is not recommendable for TSP.

(iv) For the TSP, it is direct to see that the greedy function introduced, does not capture the structure/essence of the problem. An alternative way would be to define the subsets $V_{left}, V_{right}$ using a distance criterion for the metric TSP (MS). Namely, locate a pair of vertices $u, v \in V$ satisfying $\|u - v\| = \text{diameter}(V)$; then define $V_{left}$ as the half of vertices closer to $u$ (breaking ties with the index) and $V_{right} = V - V_{left}$. However, the following must be observed. First, this approach can only be used in MS. Second, when implemented its results are no better than those presented in [20, 34, 35] and [27].

(v) It must be also be reported that, without parallel implementation, Divide-and-Conquer is not recommendable for d-KP and/or BPP, due to the involved computational time.

(vi) Although the Divide-and-Conquer method has proved to be useful for d-KP, its extension to other generalized Knapsack Problems may not be direct. It must be stressed that the tightness of the problem (introduced in Definition 3) is strongly correlated with the performance of the method for d-KP. Analogously, other generalized Knapsack Problems have to be studied, as it was done here and observe the impact of those parameters containing structural information of the problem.

(vii) Further studies will be conducted in future work, to establish the average performance of the Divide-and-Conquer method for d-KP and BPP, from the analytical point of view. Adequate probabilistic models and spaces should be introduced and analyzed.

Acknowledgments

The Author wishes to thank Universidad Nacional de Colombia, Sede Medellín for supporting the production of this work through the project Hermes 54748 as well as granting access to Gauss Server, financed by “Proyecto Plan 150x150 Fomento de la cultura de evaluación continua a través del apoyo a planes de mejoramiento de los programas curriculares” (gauss.medellin.unal.edu.co), where the numerical experiments were executed.

References

[1] F. A. Morales, J. A. Martínez, Analysis of divide-and-conquer strategies for the 0-1 minimization problem, Journal of Combinatorial Optimization 40 (1) (2020) 234 – 278.

[2] R. Loulou, E. Michaelides, New greedy-like heuristics for the multidimensional 0-1 knapsack problem, Oper. Res. 27 (1979) 1101–1114.

[3] A. Fréville, G. Plateau, Heuristics and reduction methods for multiple constraints 0-1 linear programming problems, European Journal of Operational Research 24 (1986) 206–215.

[4] G. Diubin, A. Korbut, Greedy algorithms for the minimization knapsack problem: Average behavior, Journal of Computer and Systems Sciences International 47 (1) (2008) 14–24.

[5] A. M. Frieze, M. Clarke, et al., Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses, European Journal of Operational Research 15 (1) (1984) 100–109.

[6] O. Oguz, M. Magazine, A polynomial time approximation algorithm for the multidimensional 0/1 knapsack problem, Univ. Waterloo Working Paper.

[7] B. Gavish, H. Pirkul, Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality, Mathematical Programming 31 (1985) 78–105.

[8] A. Volgenant, J. Zoon, An improved heuristic for multidimensional 0-1 knapsack problems, Journal of the Operational Research Society 41 (1990) 963–970.

[9] P. C. Chu, J. E. Beasley, A genetic algorithm for the multidimensional knapsack problem, Journal of Heuristics 4 (1998) 63–86.

[10] X. Lai, J.-K. Hao, D. Yue, Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem, European Journal of Operational Research 274 (1) (2019) 35–48. doi:10.1016/j.ejor.2018.10.000

URL https://ideas.repec.org/a/eee/ejores/v274y2019i1p35-48.html

[11] W. F. de la Vega, G. S. Lueker, Bin packing can be solved within $1 + \epsilon$ in linear time, Combinatorica 1 (1981) 349–355.
[12] N. Karmarkar, R. M. Karp, An efficient approximation scheme for the one-dimensional bin-packing problem, in: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), 1982, pp. 312–320. doi:10.1109/SFCS.1982.61

[13] A. C.-C. Yao, New algorithms for bin packing. J. ACM 27 (2) (1980) 207–227. doi:10.1145/322186.322187

[14] A. van Vliet, An improved lower bound for on-line bin packing algorithms. Information Processing Letters 43 (5) (1992) 277–284. doi:https://doi.org/10.1016/0020-0190(92)90223-1

[15] C. C. Lee, D. T. Lee, A simple on-line bin-packing algorithm. J. ACM 32 (3) (1985) 562–572. doi:10.1145/3828.3833

[16] E. G. Coffman, M. R. Garey, D. S. Johnson, Approximation Algorithms for Bin-Packing — An Updated Survey. Springer Vienna, Vienna, 2004, pp. 39–106. doi:10.1007/978-3-7908-4338-4_3

[17] T. Fischer, P. Merz, Reducing the size of traveling salesman problem instances by fixing edges, in: C. Cotta, J. van Hemert (Eds.), Evolutionary Computation in Combinatorial Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 72–83.

[18] C. Walshaw, A multilevel approach to the travelling salesman problem, Oper. Res. 50 (2002) 862–877.

[19] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Annals of Operations Research 131 (2004) 325–372.

[20] K. Ishikawa, I. Suzuki, M. Yamamoto, M. Furukawa, Solving for large-scale traveling salesman problem with divide-and-conquer strategy, SCIS & ISIS 2010 (2010) 1168–1173. doi:10.14864/softscis.2010.0.1168.0

[21] B. R. C. V. Applegate, David, On the solution of traveling salesman problems. Documenta Mathematica (1998) 645–656.

[22] D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems. Math. Program. 97 (1-2) (2003) 91–153.

[23] G. Dantzig, R. Fulkerson, S. Johnson, Solution of a large-scale traveling-salesman problem. Journal of the Operations Research Society of America 2 (4) (1954) 393–410.

[24] S. Lin, B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (1973) 498–516.

[25] D. S. Johnson, Local optimization and the traveling salesman problem, in: Proceedings of the 17th International Colloquium on Automata, Languages and Programming, ICALP '90, Springer-Verlag, Berlin, Heidelberg, 1990, p. 446–461.

[26] W. Cook, P. Seymour, Tour merging via branch-decomposition. INFORMS Journal on Computing 15 (3) (2003) 233–248.

[27] D. Applegate, W. Cook, A. Rohe, Chained Lin-Kernighan for large traveling salesman problems. INFORMS Journal on Computing 15 (1) (2003) 82–92.

[28] P. Billingsley, Probability and Measure, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, Inc., New York, NY, 1995.

[29] S. K. Thompson, Sampling, Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., New York, 2012.

[30] R. Mansini, M. G. Speranza, Corral: An exact algorithm for the multidimensional knapsack problem. INFORMS J. on Computing 24 (3) (2012) 399–415. doi:10.1287/ijoc.110.0460

[31] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Inc., USA, 1990.

[32] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, 5th Edition, Springer Publishing Company, Incorporated, 2012.

[33] D. Bertsimas, J. N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific and Dynamic Ideas, LLC, Belmont, MA, 1997.

[34] B. Fritzke, P. Wilke, Flexmap-a neural network for the traveling salesman problem with linear time and space complexity, in: [Proceedings 1991 IEEE International Joint Conference on Neural Networks, 1991, pp. 929–934 vol.2. doi:10.1109/INNS.1991.170519

[35] D. S. Johnson, L. A. McGeoch, Experimental Analysis of Heuristics for the STSP. Springer US, Boston, MA, 2007, pp. 369–443. doi:10.1007/0-306-48213-4_9