Variable Neighborhood Search applied to Multi-way Number Partitioning Problem

Alexandre Frias Faria[1] Sérgio Ricardo de Souza[1]
Carlos Alexandre Silva[2]

Abstract

This paper presents an algorithm for the optimization version of Multi-Way Number Partitioning Problem (MWNPP). This problem consists in distributing the elements of a given sequence into k disjoint subsets so that the sums of each subset elements fits in the shortest interval. The metaheuristics Variable Neighborhood Search (VNS), adapted for solving the MWNPP, has a good performance over instances less than six subsets. A comparative study with two algorithms of the literature (Karmarkar-Karp Heuristic and Longest Processing Time) is carried out, using randomly generated instances and objective functions values. The statistical tests shows that the results of the VNS proposed are significantly better than constructive methods and improved literature heuristics.

Keywords: Combinatorial Optimization, Multi-way Number Partitioning Problem, Metaheuristic.

[1] CEFET-MG, Av. Amazonas 7675, 30510-000 – Nova Gameleira – Belo Horizonte – MG – Brasil
Email: `alexandrefrias1@hotmail.com`, `sergio@dppg.cefetmg.br`
[2] IFMG, Av. Serra da Piedade 299, 34515-640 – Morada da Serra – Sabará – MG – Brasil
Email: `carlos.silva@ifmg.edu.br`

1 Introduction

In this article, a partition of a set X is a collection of subsets, two to two disjoints, whose union forms X. A k-partition is a partition having exactly k non-empty subsets. The subsets belonging to the partition are called parts. The set $\mathbb{Z}_+$ indicates strictly positive integers numbers and the notation $I_m = \{y \in \mathbb{Z} \ : \ 1 \leq y \leq m\}$ denotes the set of all integers between 1 and m.

This paper addresses the Multi-Way Number Partitioning Problem, in the sequel named MWNPP, first level generalization of the Two-Way Number Partitioning Problem (TWNPP). Let V be a numerical sequence. In the MWNPP, the objective is to find a k-partition of the indexes of V, so that the sums of the elements of each part are as close as possible to each other. This comes down to having the largest sum part as close to the smallest part as possible. An immediate conclusion is that, in relation to TWNPP, the MWNPP expands the number of parts in which the elements of the V sequence must be distributed.

There is an extensive literature on TWNPP and its variations. It is a classical Combinatorial Optimization problem, formally listed in [5] as one of the basic NP-Complete problems. The MWNPP arises explicitly in an article dealing with the analysis of a constructive heuristic called Differencing Method, better known as Karmarkar-Karp Heuristic (KKH), proposed by [4]. This heuristic seeks to divide the largest numbers of the sequence V into distinct parts, inserting, in the set of unallocated elements, the differences between the elements removed, as long as it is not empty. In [10], an approximation ratio given by $[4/3 - 1/(3(k-1))]$ is shown for KKH. This result imposes a limited error interval for the values obtained with KKH, being similar to the ratio found for the Longest Processing Time (LPT) algorithm, proposed in [3].

According to [1], the MWNPP is a very difficult problem to be solved by general-use metaheuristics, like Genetic Algorithms, Simulated Annealing and others. In many cases, these methods lose in terms of computational time and in performance for the KKH and even for the LPT. The construction of exact algorithms is proposed in [6], where a Backtrack procedure is performed in constructive heuristics, like LPT and KKH, called Complete Greedy Algorithm (CGA) and Complete Karmarkar-Karp Algorithm (CKKA), respectively. The first improvement in these works happens with the algorithm Recursive Number Partitioning, proposed by [7]. The second improvement is the contribution of [13], in which a new data structure applied to the work of [7] is proposed, speeding up the search in the Karmarkar-Karp Tree. Through successive MWNPP conversions from a $(k-1)$-partition to a k-partition, an algorithm based on smaller subproblems resolution is proposed in [12]. Currently, the state of the art for MWNPP is the Sequential Number Partitioning algorithm [8] and the Cached Iterative Weakening algorithm, [14].

This article presents an adaptation of the VNS method for the MWNPP solution and a comparison of this solution with that performed by two constructive methods of the literature (KKH and the intensified LPT). The main reason for applying the VNS to the MWNPP is the set of good results found in [9] for the solution of the Multidimensional Two-way Number Partitioning Problem (MTWNPP). The current article is organized as follows: Section 2 presents the statement and mathematical model of the problem treated; Section 3 shows the proposed VNS; Section 4 presents the tests performed for the comparison between the algorithms; Section 5 ends the paper, showing the conclusions.

2 Problem Setup

The MWNPP treated here is the version originally addressed in [4]. The input is a $V = \{v_1, v_2, \ldots, v_n\}$ sequence of positive integer numbers and the output is a k-partition of the indexes of V, where k is a positive integer number. The k-Number Partitioning Problem (Multi-way Number Partitioning Problem) consists of finding a k-partition of the indexes of V, in the form $\{A_1, A_2, \ldots, A_k\}$, that minimize the function:

$$f(\{A_1, A_2, \ldots, A_k\}) = \max_{j'} \left\{ \sum_{i \in A_{j'}} v_i \right\} - \min_{j} \left\{ \sum_{i \in A_j} v_j \right\} \tag{1}$$

The Integer Linear Optimization mathematical model of this problem can be described as follows:

$$\begin{aligned}
\min \quad & t_2 - t_1 && (2)\\
\text{sub. to} \quad & t_1 \le \sum_{i=1}^{n-j+1} v_i x_{ji} \le t_2, \quad \forall j \in I_k && (3)\\
& \sum_{j=1}^{\min\{n-i+1,k\}} x_{ji} = 1, \quad \forall i \in I_n && (4)\\
& t_1, t_2 \in \mathbb{Z}_+ && (5)\\
& x_{ji} \in \{0,1\}, \quad \forall (j,i) : i + j \le n+1 && (6)
\end{aligned}$$

The expression (2) is the objective function to be minimized, and represents the size of the interval that contains all the k sums of the elements of the parts. The k inequalities (3) show that the model guarantees that each part is non-empty, since $t_1 > 0$, due to expression (6), and also that all parts are contained in the range $[t_1, t_2]$. The n equations indicated by the expression (4) ensure that the parts are disjoint and that all elements of V are allocated somewhere, once the problem is only well defined when $n \geq k$. The relation (5) shows that the limitings of the largest and smallest sum of the elements of the parts are always positive. The relation (6) indicates that $x_{ji} = 1$ if the index element i of the set V belongs to the index part j. Only pairs (j, i) such that $i+j \leq n+1$ must be defined, since the order of the parts does not change the solution. Thus, the last element v_{n-1} can only be in the parts between 1 and $l+1$ for any feasible solution of the problem. This model founds the representation of a feasible MWNPP solution as being a vector s such that $s_i = j$ if $v_i \in A_j$. It is also impossible to have $j > 1 + \max_{1 \leq l \leq i} \{s_l\}$ to avoid multiplicities of equal solutions.

3 Proposed Algorithm

The adaptation of the Variable Neighborhood Search (VNS) metaheuristic proposed here works with reallocation movements of an element between parts of the partition $\{A_1, A_2, \ldots, A_k\}$. The movement $m_{i,j}$, defined only for $i \notin A_j$, removes an index element i from the part where it is and relocates it to the index part j. Thus, if $i \in A_l$, a reallocation movement $\{A_1, A_2, ...A_l, ..., A_j, ..., A_k\} \oplus m_{i,j}$ leads to $\{A_1, A_2, ...A_l - \{i\}, ..., A_j \cup \{i\}, ..., A_k\}$. Let $s' = \{A'_1, A'_2, ..., A'_k\}$ and $s = \{A_1, A_2, ..., A_k\}$. The neighborhoods $N_1(s)$, $N_2(s)$ e $N_3(s)$ are defined, in consequence, by compositions of one, two, and three distinct movements $m_{i,j}$, as follows:

$$N_1(s) = \{s' \ : \ s' \leftarrow s \oplus m_{i,j}, \ \forall (i,j) \in I_n \times I_k\} \tag{7}$$
$$N_2(s) = \{s' \ : \ s' \leftarrow s \oplus m_{i,j} \oplus m_{i',j'}, \ \forall (i,j) \in I_n \times I_k\} \tag{8}$$
$$N_3(s) = \{s' \ : \ s' \leftarrow s \oplus m_{i,j} \oplus m_{i',j'} \oplus m_{i'',j''}, \ \forall (i,j) \in I_n \times I_k\} \tag{9}$$

and are such that $N_l(s) \cap N_j(s) = \emptyset \quad \forall i \neq j$.

The Algorithm 1 describes the implemented VNS. This implementation follows the general structure for this metaheuristic, introduced in [11]. The initial solution is the worst possible for the instance, being the $k-1$ smallest elements of V defining, each one, a part $A_j = \{v_i\} \quad \forall j \in \{1, 2, ...k-1\}$. The perturbation in the current solution is a random movement $m_{i,j}$. The local search uses the Best Improvement heuristic to select the neighbor that causes the lowest decrease of the objective function and updates it as a current solution.

Algorithm 1 Adapted VNS algorithm

```
1: function VNS($s$, $f()$)                                          ▷ Initial solution.
2:     $r \leftarrow 1$                                              ▷ Initial $N_r(s)$.
3:     $s' \leftarrow s$
4:     while $cont < Itermax$ or $gap(s, s') < 0.9$ do   ▷ Stopping criterion by the improvement or iterations
5:         $s' \leftarrow \arg\min_{N_r(s)} f(s)$                      ▷ Best Improvement
6:         if $f(s') < f(s)$ then
7:             $s \leftarrow s'$
8:         else
9:             $r \leftarrow (r + 1)\ mod\ 3$                          ▷ Neighborhood exchange
10:        end if
11:        $cont \leftarrow cont + 1$
12:    end while
13:    return $s$
14: end function
```

4 Experimental Results

The tested algorithms (adapted VNS, Lpt_1 e KKH) were implemented in `C++` language. The experiments were performed on a computer with CPU Intel Core i3-M330, 2.13GHz, 3GB of RAM and Ubuntu 16.04 (32 bits) operational system, using the compiler `g++` version 5.4 for their execution. The instances for the experiments were generated randomly. The generated V sequences have n elements, with $n \in \{100, 200, 400, 800\}$, and the elements are integers, varying from 0 to 999999999999, with 12 digits and uniform distribution. For this end, a number between 0 and 9, inclusive, is randomly generated for 12 consecutive times. This process is repeated n times for the generation of a test instance.

The objective of the computational experiments of this article is to perform a comparison between the result found by the constructive heuristics KKH, proposed by [6] and Lpt_1, an intensification of the LPT proposed by [2], and the result obtained by the Adapted VNS metaheuristic, showed in Algorithm 1. The measures for the comparison of the results are the objective function values in each algorithm and the measure $gap(B, A) = \frac{z(A)-z(B)}{z(A)}.100\%$, which shows how much the response of the algorithm B differs in percentage from the response of the algorithm A, where $z(A)$ and $z(B)$ are their respective objective function values. The spent computational time is not subject to comparison, since the algorithms have complexities between $O(n.log(n))$, for KKH and Lpt_1, and $O(n^3)$, for the VNS. Table 1 presents the computational results. Note that the order of the algorithms at the input of the $gap()$ function was carefully chosen to maintain most of the positive values and to facilitate the verification of the data and hypotheses necessary to perform the t-test.

Table 1
Comparison of the computational results from KKH, Lpt_1 and VNS algorithms.

	KKH vs VNS - $gap(VNS, KKH)$				Lpt_1 vs VNS - $gap(VNS, Lpt\_1)$			
inst/ k	k=3	k=4	k=5	k=6	k=3	k=4	k=5	k=6
inst1	92,40%	98,17%	97,08%	98,45%	47,07%	66,94%	63,70%	64,51%
inst2	99,82%	99,64%	99,40%	97,35%	81,22%	96,59%	94,55%	70,72%
inst3	99,93%	99,71%	99,69%	97,13%	94,73%	52,76%	64,73%	**-148,46%**
inst4	95,38%	98,84%	98,92%	96,42%	92,40%	96,86%	80,19%	21,44%
inst5	99,99%	99,98%	99,94%	99,54%	92,66%	93,35%	92,21%	80,59%
inst6	99,98%	99,94%	99,92%	99,75%	98,37%	98,83%	96,32%	28,94%
inst7	99,66%	99,90%	98,78%	99,18%	97,80%	91,02%	78,59%	73,76%
inst8	99,75%	99,97%	99,65%	99,87%	95,11%	81,52%	72,91%	93,46%
inst9	99,82%	99,95%	99,89%	99,74%	96,79%	93,91%	77,57%	87,84%
inst10	99,99%	99,93%	99,77%	99,60%	99,50%	93,08%	92,25%	79,08%
inst11	100,00%	100,00%	99,96%	99,80%	99,08%	97,96%	88,96%	44,40%
inst12	100,00%	99,98%	99,87%	99,12%	99,82%	97,60%	94,93%	41,95%
inst13	100,00%	99,99%	99,58%	99,88%	98,89%	99,14%	53,08%	95,32%
inst14	99,98%	99,70%	99,48%	99,82%	99,61%	98,16%	88,16%	95,83%
inst15	99,36%	99,99%	99,91%	98,72%	97,42%	98,74%	92,82%	80,50%
inst16	100,00%	100,00%	99,99%	99,97%	99,91%	99,71%	97,57%	87,18%
inst17	99,99%	99,97%	99,89%	99,70%	97,79%	97,43%	92,41%	22,17%
inst18	99,99%	100,00%	99,92%	99,95%	98,34%	99,72%	81,09%	87,06%
inst19	99,99%	99,98%	99,90%	99,90%	99,28%	97,00%	78,36%	89,19%
inst20	100,00%	100,00%	99,99%	99,96%	99,65%	99,78%	96,85%	93,69%

Table 2
Paired two-sample t-tests in samples of objective functions values

p-values $H_1: \ f_{VNS} < f_{KKH}$				p-values $H_1: \ f_{VNS} < f_{Lpt\_1}$			
k=3	k=4	k=5	k=6	k=3	k=4	k=5	k=6
0,010%	0,000%	0,000%	0,000%	0,111%	0,060%	0,020%	0,020%

The values of $gap() = 100\%$ are due to rounding.

Table 3
One-sample t-test in *gap* of the table 1 with $(1 - \alpha) = 0.99$.

p-values $H_1: \ gap(VNS, KKH) > 0.9$				p-values $H_1: \ gap(VNS, Lpt\_1) > 0.9$			
k=3	k=4	k=5	k=6	k=3	k=4	k=5	k=6
0,000%	0,000%	0,000%	0,000%	5,444%	17,845%	98,489%	99,355%

Table 2 shows the p-values of the statistical tests using objective function values f_{alg} as samples. In all 8 tests it is possible to state, with 99% confidence, that the results found by the Algorithm 1 are smaller than the results of the KKH and the Lpt_1 algorithms. After confirming the superiority of the

Algorithm 1, the tests shown in Table 3 expose the p-values of the statistical tests using values of $gap()$ as samples. In the first 4 tests it is possible to state, with 99% confidence, that the results from Algorithm 1 are at least 10 times lower than those of KKH. The same has not happened in the last 4 tests, regarding Lpt_1. For $k = \{3, 4, 5\}$, the tests would reject H_0 only if $H_1 :\ gap(VNS, Lpt\_1) > 0,75$. The Algorithm 1 did not overcome Lpt_1 at instance inst3 with $k = 6$. In this case, the response of Lpt_1 is about 2.48 times lower than that of Algorithm 1. Even if this value were discarded, it would not be possible to affirm the superiority of Algorithm 1 in a test in which $H_1 :\ gap(VNS, Lpt\_1) > 0,75$, as in the cases for $k = \{3, 4, 5\}$.

5 Conclusions and Future Works

This article presents a proposal for adapting the VNS metaheuristic for the MWNPP solution and a comparison of the results obtained with this adaptation with results from constructive algorithms in the literature. The algorithms were tested with randomly generated instances having uniformly distributed elements. The results show that the adaptation of the VNS metaheuristic overcame the results of the constructive algorithms in the absolute majority of the instances. As future work, it is proposed the application of the VNS metaheuristic in an implicit enumeration method, as well as the accomplishment of a test of statistical significance having, as measures, the solution time and the memory usage.

References

[1] Gent, I. P. and T. Walsh, *Analysis of heuristics for number partitioning*, Computational Intelligence **14** (1998), pp. 430–451.

[2] Graham, R. L., *Bounds for certain multiprocessing anomalies*, The Bell System Technical Journal **XLV** (1966), pp. 1563–1581.

[3] Graham, R. L., *Bounds on multiprocessing timing anomalies*, SIAM Journal on Applied Mathematics **17** (1969), pp. 416–429.

[4] Karmarkar, N. and R. M. Karp, *The differencing method of set partition*, Report UCB/CSD 81/113, Computer Science Division, University of California, Berkeley, CA (1982).

[5] Karp, R. M., *Reducibility among combinatorial problems*, in: R. E. Miller, J. W. Thatcher and J. D. Bohlinger, editors, *Proceedings of a Symposium on the*

Complexity of Computer Computations (1972), pp. 85–103.

[6] Korf, R. E., *A complete anytime algorithm for number partitioning*, Artificial Intelligence **106** (1998), pp. 181–203.

[7] Korf, R. E., *Multi-way number partitioning.*, in: *IJCAI*, Citeseer, 2009, pp. 538–543.

[8] Korf, R. E., E. L. Schreiber and M. D. Moffitt, *Optimal sequential multi-way number partitioning*, in: *International Symposium on Artificial Intelligence and Mathematics (ISAIM-2014)*, Fort Lauderdale, FL, USA, 2013.

[9] Kratica, J., J. Kojić and A. Savić, *Two metaheuristic approaches for solving multidimensional two-way number partitioning problem*, Computers & Operations Research **46** (2014), pp. 59 – 68.

[10] Michiels, W., J. Korst, E. Aarts et al., *Performance ratios for the karmarkar-karp differencing method*, Electronic Notes in Discrete Mathematics **13** (2003), pp. 71–75.

[11] Mladenović, N. and P. Hansen, *Variable neighborhood search*, Computers & Operations Research **24** (1997), pp. 1097 – 1100.

[12] Moffitt, M. D., *Search strategies for optimal multi-way number partitioning*, in: *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, AAAI Press, 2013, pp. 623–629.

[13] Pedroso, J. P. and M. Kubo, *Heuristics and exact methods for number partitioning*, European Journal of Operational Research **202** (2010), pp. 73–81.

[14] Schreiber, E. L. and R. E. Korf, *Cached iterative weakening for optimal multi-way number partitioning*, in: *Proceedings of the 28th Annual Conference on Artificial Intelligence (AAAI-14)*, Quebec City, Canada, 2014, pp. 2738–2745.