Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 346 (2019) 437–447](https://doi.org/10.1016/j.entcs.2019.08.039)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Variable Neighborhood Descent Branching applied to the Multi-Way Number Partitioning Problem

Alexandre Frias Faria[1](#_bookmark0) S´ergio Ricardo de Souza[1](#_bookmark0) Elisangela Martins de Sa´[1](#_bookmark0)

*Centro Federal de Educac¸˜ao Tecnol´ogica de Minas Gerais*

*Av. Amazonas 7675, 30510-000 – Nova Gameleira – Belo Horizonte – MG – Brazil*

Carlos Alexandre Silva[2](#_bookmark0)

*Instituto Federal de Educa¸c˜ao Tecnol´ogica de Minas Gerais*

*Av. Serra da Piedade 299, 34515-640 – Morada da Serra – Sabar´a – MG – Brasil*

**Abstract**

This paper presents an application of the Variable Neighborhood Descent Branching method to solve the Multi-Way Number Partitioning Problem. This problem consists of distributing the elements of a given sequence into *k* disjoint subsets such that the sums of each subset elements fit in the shortest interval. It shows a new method to decompose the MWNPP in *n−*1 subproblems using local branching constraints. This decomposing justifies the neighborhood structure used in the proposed algorithm. The study of parameter settings defines the operation of the proposed algorithm. The results shows that there is no statistically significant difference of objective value between proposed algorithm and mathematical model solved by CPLEX, but the time used by both methods are significantly different.

*Keywords:* Combinatorial Optimization, Multi-Way Number Partitioning Problem, Variable Neighborhood Descent Branching, Matheuristics.

# Introduction

A partition of a set *X* is a collection of mutually disjoint subsets whose union forms *X*. A *k*-partition is a partition with exactly *k* non-empty subsets. In this article, the subsets belonging to the partition are called parts, the set Z+ denotes

1 Email: [alexandrefrias1@hotmail.com](mailto:alexandrefrias1@hotmail.com), [sergio@dppg.cefetmg.br](mailto:sergio@dppg.cefetmg.br), [elisangelamartins@cefetmg.br](mailto:elisangelamartins@cefetmg.br)

2 Email: [carlos.silva@ifmg.edu.br](mailto:carlos.silva@ifmg.edu.br)

<https://doi.org/10.1016/j.entcs.2019.08.039>

1571-0661/© 2019 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

the set of strictly positive integers. Furthermore, the notation *Im* = *{y ∈* Z : 1 *≤*

*y ≤ m}* represents the closed set of integers between 1 and *m*.

The Two-Way Number Partitioning Problem (TWNPP) consists of finding a 2-partition for the indexes of a given sequence *V* . The purpose of the problem is to minimize the difference between the sums of elements in distinct parts. A gener- alization of TWNPP is the Multi-Way Number Partitioning Problem (MWNPP). In this problem, the number of parts *k* in which the sequence elements *V* are to be distributed is fixed. Let the weight *g*(*Aj*) be given by the sum of the elements whose indexes are contained in the part *Aj*. Given a numerical sequence *V* , the

goal is to find a *k*-partition for its indexes, so that the set of weights *{g*(*Aj*)*}k* be

*j*=1

contained in the shortest possible interval.

TWNPP is formally listed in [[7](#_bookmark22)] as one of the basic NP-complete problems. There are a number of equivalences demonstrated between TWNPP and other NP- complete problems. On the other hand, MWNPP appears originally in an article about the analysis of a constructive heuristic called Differencing Method, better known as Karmarkar-Karp Heuristic (KKH), proposed in [[6](#_bookmark21)]. According to [[4](#_bookmark18)], MWNPP is a very difficult problem to solve using general purpose meta-heuristics, such as Genetic Algorithms, Simulated Annealing and others. In many cases, these methods lose in computational time and performance for constructive heuristics such as KKH and even for the Longest Processing Time heuristic (LPT), proposed by [[5](#_bookmark20)]. The construction of exact algorithms for the solution of these problems is proposed in [[8](#_bookmark23)]. For this, a Backtrack procedure is performed in constructive heuristics. When the LPT heuristic is used during the enumeration, the Complete Greedy Algorithm (CGA) is produced; if KKH is used, the Complete Karmarkar-Karp Algorithm (CKK) is generated. The first improvement to be proposed in these algorithms occurs with the algorithm Recursive Number Partitioning (RNP), proposed by [[9](#_bookmark24)]. The second improvement is due to [[12](#_bookmark27)], in which a new data structure applied to the CKK algorithm is proposed, speeding up the search in the Karmarkar-Karp Tree. Through successive MWNPP conversions of a (*k −* 1)-partition to a *k*-partition, [[11]](#_bookmark26) proposes an algorithm based on solving smaller subproblems. Currently, the state of the art for MWNPP is the Sequential Number Partitioning algorithm, presented in [[10](#_bookmark25)], and the Cached Iterative Weakening algorithm, shown in [[14](#_bookmark28)]. A complete and highly relevant analysis of these algorithms is found in [[13](#_bookmark29)].

Given a mathematical model for the MWNPP, a branching technique by inser- tion of Local Branching Constraints is proposed in [[3](#_bookmark19)]. These constraints are defined from an initial solution *xs*, and determine how close or far a feasible solution can be of *xs*. This technique allowed the proposition of many exact and approximate meth- ods for the solution of combinatorial optimization problems. One of these methods is the Variable Neighborhood Descent Branching (VNDB), described in [[1](#_bookmark16)], which defines its neighborhoods using Local Branching Constraints.

This article addresses an adaptation of the VNDB method to the MWNPP us- ing the constructive heuristic LPT as an initial solution. The ability to exclude regions already exploited from the search space by simply adding constraints to the mathematical model justifies the application of VNDB to solve this problem.

The analysis of the computational time spent and the upper bound for each in- stance shows that VNDB, with a reasonable number of neighborhoods, finds results as good as the standard CPLEX solver and, in addition, in a shorter run time. The article is organized as follows. Section [2](#_bookmark1) presents the problem addressed and states a mathematical model for MWNPP. Section [3](#_bookmark5) introduces an analysis of local branching constraints. Section [4](#_bookmark10) addresses the proposed VNDB method and evalu- ates the particularity of its neighborhood in rings. Following, Section [5](#_bookmark13) presents the computational tests performed for the comparison between the algorithms. Finally, Section [6](#_bookmark14) concludes the paper, including a critique about the found results and the proposed scope.

# Problem Statement

MWNPP treated here is the version originally addressed in [[6](#_bookmark21)]. Its input is a sequence *V* and its output is a *k*-partition of the indexes of *V* .

**Definition 2.1** Let *V* = *{v*1*, v*2*,..., vn}* be a sequence of positive integers, and *k* a positive integer. Find a *k*-partition of the *V* indexes, in the form *{A*1*, A*2*,..., Ak}*, that minimizes the function:

*f* (*{A*1*, A*2*,..., Ak}*)= max*{g*(*Aj′* )*}−* min*{g*(*Aj*)*}.* (1)

*j′ j*

For instance, the sequence *V* = *{*11*,* 25*,* 13*,* 34*,* 89*,* 65*,* 43*,* 96*,* 56*,* 87*}* may be parti- tioned in the ways shown in Table [1](#_bookmark2) for the values *k ∈ {*3*,* 4*,* 5*,* 6*}*. The feasible and optimal values of the objective function for these cases are presented in Table [2.](#_bookmark3)

feasible optimal

*k* =3 *{*96*,* 43*,* 34*}, {*89*,* 56*,* 25*}, {*87*,* 65*,* 13*,* 11*} {*89*,* 87*}, {*25*,* 13*,* 34*,* 43*,* 56*}, {*11*,* 65*,* 96*}*

v ,

173

v ,

170

v ,

176

v ,

176

v ,

171

v ,

172

*k* =4 *{*96*,* 25*,* 13*}, {*89*,* 34*}, {*87*,* 43*}, {*65*,* 56*,* 11*} {*25*,* 13*,* 89*}, {*43*,* 87*}, {*34*,* 96*}, {*11*,* 65*,* 56*}*

v ,

134

v ,

123

v ,

130

v ,

132

v ,

127

v ,

130

v ,

130

v ,

132

*k* =5 *{*11*,* 89*,* 13*}, {*25*,* 87*}, {*96*}, {*34*,* 65*}, {*43*,* 56*} {*11*,* 89*}, {*25*,* 87*}, {*13*,* 96*}, {*34*,* 65*}, {*43*,* 56*}*

v ,

113

v ,

112

v ,

96

v ,

99

v ,

99

v ,

100

v ,

112

v ,

109

v ,

99

v ,

99

*k* =6 *{*96*}, {*89*}, {*87*}, {*11*,* 13*,* 65*}, {*56*,* 25*}, {*43*,* 34*} {*96*}, {*89*}, {*87*}, {*11*,* 25*,* 43*}, {*13*,* 65*}, {*34*,* 56*}*

v ,

96

v ,

89

v ,

87

v ,

89

v ,

81

v ,

77

v ,

96

v ,

89

v ,

87

v ,

79

v ,

78

v ,

90

Table 1

Solutions of MWNPP for *k ∈ {*3*,* 4*,* 5*,* 6*}*.

Table 2 Objective function values.

|  |  |  |
| --- | --- | --- |
|  | feasible | optimal |
| *k* = 3 | 176 - 170 = 6 | 176 - 171 = 5 |
| *k* = 4 | 134 - 123 = 11 | 132 - 127 = 5 |
| *k* = 5 | 113 - 96 = 17 | 112 - 99 = 13 |
| *k* = 6 | 96 - 77 = 19 | 96 - 78 = 18 |

MWNPP can be modeled using binary variables *xji* to indicates if the *i*-th term of the sequence *V* belongs to the part *j* (*xji* = 1) or not (*xji* = 0). Only pairs (*j, i*)

outside the lower right triangle must be set to avoid multiplicities. Using this set of variables, a mixed integer programming (MIP) formulation for MWNPP can be given as:

min *t*2 *− t*1 (2a)

*n*

Σ

st. *t*1 *≤ vixji ≤ t*2*, ∀j ∈ Ik* (2b)

*i*=*j*

min*{i,k}*

Σ*j*=1

*xji* = 1*, ∀i ∈ In* (2c)

*t*1*, t*2 *∈* R+ (2d)

*xji ∈ {*0*,* 1*}, ∀*(*j, i*): *i ≥ j* (2e)

The objective function ([2a](#_bookmark4)) minimizes the size of the interval containing

*{g*(*Aj*)*}k*

*j*=1

. Constraints ([2b](#_bookmark4)) guarantees that each part is non-empty, since *t*1 *>* 0,

and that the weight associated to each part are contained in the interval [*t*1*, t*2]. Constraints ([2c](#_bookmark4)) guarantee that the parts are disjoint and that all elements of *V* are allocated to some part, since the problem is only well defined when *n ≥ k*. Finally, constraints ([2d](#_bookmark4)) ensure that the constraints of max*j{g*(*Aj*)*}* and min*j{g*(*Aj*)*}* are always positive. This MIP model were first introduced in [[2](#_bookmark17)], with a little difference in relation to the variables excluded.

The solution of this model implies the solution of MWNPP, but there is still a surjective correspondence between feasible solutions of the mathematical model and the set of *k*-partitions.

# Analysis of Local Branching Constraints

Local Branching constraints define the Hamming distance between a given solu- tion and a feasible solution of the problem by counting the number of variables with different values [[3](#_bookmark19)]. Consider an incumbent solution *xs* for the mathematical model

([2](#_bookmark4)). Let *B* be the set of pairs of indexes (*j, i*) such that *xs*

*ji*

= 1. Local Branching

constraints are defined by the following expressions:

Δ(*xs, x*) := Σ (1 *− xji*) + Σ *xji ≤ R* (3a)

*ji∈B*

*ji/∈B*

Δ(*xs, x*) := Σ (1 *− xji*) + Σ *xji* = *R* (3b)

*ji∈B*

*ji/∈B*

Δ(*xs, x*) := Σ (1 *− xji*) + Σ *xji ≥ R* (3c)

*ji∈B*

*ji/∈B*

The constraint ([3a](#_bookmark6)) means that a feasible solution must be at a Hamming distance less than or equal to *R* with respect to the solution *xs*. Thus, the neighborhood *NR*(*xs*) is a disk of radius *R* and center *xs*. In the constraint ([3b](#_bookmark6)), only solutions with Hamming distance exactly equal to *R* in relation to *xs* are allowed. This neighborhood structure is a degenerate ring with center at *xs* and internal and external radius equal to *R*. Finally, the constraint ([3c](#_bookmark6)) considers all the space

outside the radius disk *R* and center *xs* as space of feasible solutions.

It is important to note that Local Branching constraints do not always define the exact neighborhood structure, because its shape will depend on the intersection with the feasible region of the mathematical model of the problem studied.

Let *x*1 and *x*2 be two distinct feasible solutions of model ([2](#_bookmark4)). The first column

11

of the solution matrix always has the values *x*1

11

= 1 and *x*2

= 1. Therefore, there

are *n −* 1 columns left for the variables to assume different values. From the second column forwards, an increase of at most two units per column at the total Hamming distance can be obtained since, due to the constraints ([2c](#_bookmark4)), there is only one *j* such that *xji* = 1 for each *i* column. Thus, the Hamming distance between two feasible solutions of the mathematical model ([2](#_bookmark4)) is always an even number. Therefore, it can be concluded that:

*∀x*1*, x*2 *∃r ∈ In−*1 : Δ(*x*1*, x*2)= 2*r* (4)

and, as a consequence of ([4](#_bookmark7)), the maximum Hamming distance between two feasible solutions can be given as:

max*{*Δ(*x*1*, x*2)*}* = 2(*n −* 1)*.* (5)

Therefore, there is a decomposition of MWNPP into subproblems with con- straints ([3b](#_bookmark6)). Using the equation ([4](#_bookmark7)), a subproblem *Pr*(*xs*) resulted by adding the constraint Δ(*xs, x*)= 2*r* to the model ([2](#_bookmark4)) is given as:

(*Pr*(*xs*)) min *t*2 *− t*1 (6a)

*n*

Σ

st. *t*1 *≤ vixji ≤ t*2*, ∀j ∈ Ik* (6b)

*i*=*j*

min*{i,k}*

Σ*j*=1

*xji* = 1*, ∀i ∈ In* (6c)

Σ (1 *− xji*) + Σ *xji* = 2*r* (6d)

*ji∈B*

*ji/∈B*

*t*1*, t*2 *∈* R+ (6e)

*xji ∈ {*0*,* 1*}, ∀*(*j, i*): *i ≥ j* (6f)

It is known that the mathematical model ([6](#_bookmark9)) only assumes feasible solutions when 1 *≤ r ≤ n −* 1 due to the equation ([5](#_bookmark8)). It is also important to note that *Pr* (*xs*) and *Pr* (*xs*) have no feasible solution in common whenever *r*1 */*= *r*2, because its distances from solution *xs* are distinct in this case. Therefore, the collection of subproblems *{Pr*(*xs*)*}n−*1 is a decomposition of the mathematical model ([2](#_bookmark4)) and, thus, a decomposition of MWNPP.

1

2

*r*=1

# Proposed Algorithm

The proposed VNDB algorithm works with the insertion and removal of Lo- cal Branching Constraints aiming, in addition to the definition of neighborhoods,

the memory of the search space already explored using mathematical expressions. Thus, let *x*1 be a feasible solution of the problem ([2](#_bookmark4)) and *x*2 be the solution of the subproblem *P*1(*x*1). The initial problem of VNDB is *P*1(*x*1). The next subproblem will be *P*2(*x*1) or *P*1(*x*2) *∩ {*Δ(*x*1*, x*) *≥* 2*}*, depending on which is the best solution, or *x*1 or *x*2. The same logic applies to each of the possibilities.

In general, method decisions are made as follows. Let *M* be a set of constraints of

type ([3c](#_bookmark6)) and (*xl*)*m*

*l*=1

a sequence of solutions found by VNDB. In the iteration *t*, the

solution of the subproblem *Pr*(*xl*)*∩M* leads to the solution *xl*+1. If the solution *xl*+1 is better than *xl*, the subproblem is updated by adding the constraint Δ(*xl, x*) *≥* 2(*r* + 1) to the set *M* and yielding the set *Mj*. This step eliminates the feasible region of the *Pr*(*xl*) subproblem. Finally, the constraint Δ(*xl, x*) = 2*r* is replaced by the constraint Δ(*xl*+1*, x*) = 2. This procedure creates a problem in the form *P*1(*xl*+1) *∩Mj*. If the solution *xl*+1 is no better than *xl*, the constraint Δ(*xl, x*)= 2*r* is changed by Δ(*xl, x*)= 2(*r* + 1), generating the subproblem *Pr*+1(*xl*) *∩ M* .

The mathematical model ([7](#_bookmark11)) shows the structure of the VNDB subproblem after

passing through *m* distinct solutions and performing *t* = Σ*m rl* iterations.

*l*=1

min *t*2 *− t*1 (7a)

*n*

Σ

s.t. *t*1 *≤ vixji ≤ t*2*, ∀j ∈ Ik* (7b)

*i*=*j*

min*{i,k}*

Σ*j*=1

*xji* = 1*, ∀i ∈ In* (7c)

Σ (1 *− xji*) + Σ *xji* = 2*rm* (7d)

*ji∈Bm*

*ji/∈Bm*

Σ (1 *− xji*) + Σ *xji ≥* 2*rl, ∀l ∈ Im—*1 (7e)

*ji∈Bl*

*ji/∈Bl*

*t*1*, t*2 *∈* R+ (7f)

*xji ∈ {*0*,* 1*}, ∀*(*j, i*): *i ≥ j* (7g)

The subproblem ([7](#_bookmark11)) has *m* more constraints than the mathematical model ([2](#_bookmark4)). The constraint ([7d](#_bookmark11)) represents the current neighborhood of the iteration. The *m−* 1 constraints ([7e](#_bookmark11)) make infeasible the search regions of the previous subproblems. The increase of *m* with the number of iterations depends on the initial solution chosen and the size of the neighborhood structure. The values *rl ∈ In—*1 define by how many neighborhoods each solution passed without being surpassed.

**Algorithm 1** VNDB

1: **function** VNDB(*x*˙, *N* ) *d* Initial solution and number of neighborhoods

2: *r →* 1 *d* Initial neighborhood

3: *t →* 1 *d* Iteration

4: **while** *t < Itermax* and *r ≤ N* **do**

5: *add* Δ(*x*˙*, x*)= 2*r*

6: *x*¨ *→ solve*() *d* Subproblem solution

7: *remove* Δ(*x*˙*, x*)= 2*r*

8: **if** *optimal* **then**

9: **if** *ƒ* (*x*¨) *≥ ƒ* (*x*˙) **then** *d* Without improved solution

10: *r → r* +1 *d* Next neighborhood

11: **else** *d* Improved solution

12: *add* Δ(*x*˙*, x*) *≥* 2(*r* + 1) *d* Exclusion of explored search space

13: *r →* 1 *d* Return to the first neighborhood

14:

*x*˙ *→ x*¨

*d* Update solution

15: **end if**

16: **else if** *ƒeasible* **then** *d* No search space reduction

17: **if** *ƒ* (*x*¨) *≥ ƒ* (*x*˙) **then**

18: *r → r* +1

19: **else**

20: *r →* 1

21:

*x*˙ *→ x*¨

22: **end if**

23: **else** *d* Infeasible or *solve*() does not find solution

24: *r* = *r* +1

25: **end if**

26: *t → t* +1

27: **end while**

28: **return** *x*˙

29: **end function**

Algorithm [1](#_bookmark12) summarizes the above explanations. The stopping criterion is the exhaustion of the *N* neighborhoods in a given solution or by the maximum number of iterations *Itermax* (or maximum time). In practice, the function *solve*() also has a time limit to resolve the VNDB subproblem. Then there must be conditional deviations for all possible solution status: *{feasible, optimal, infeasible}*.

When the method *solve*() returns a feasible but not optimal improvement so- lution, it is not possible to exclude the subproblem region because there is some unexploited space that may still contain the optimal solution to the general problem.

# Computational Experiments

The proposed VNDB algorithm and the mathematical model presented in Sec- tion [2](#_bookmark4) were coded in C++ using the Concert Technology of CPLEX 12.6 in the default configuration. The computational experiments were carried out on an Intel Core i7-3770 CPU 3.4 GHz with 8 cores and 32GB RAM running on a Ubuntu

16.04 64-bit operating system using clang compiler version 3.8. The test instances composed of a set of sequences were randomly generated as follows. Sequences with *n* integer elements with 12 digits ranging from 0000000000 to 999999999999 are generated by randomly sampling for 12 consecutive times a number between 0 and 9 using a uniform distribution.

The computational experiments are performed aiming to compare the quality of the solution *ub* and the computational time associated to the VNDB and the CPLEX solver applied in the mathematical model [2](#_bookmark4) considering a time limit of *t* = 4500 seconds. The measures used to compare the results are based on the relative error measure given by *gap*(*B, A*)= *z*(*A*)*—z*(*B*) 100%, where *z*(*A*) and *z*(*B*) corresponds to the objective function value associated to the best solution provided by algorithm *A* and algorithm *B*, respectively. This function shows that the solution provide by the algorithm *B* is better than the solution provided by the algorithm *A* when *gap*(*B, A*) *>* 0. For instance, the *gap*(*B, A*) = 90% means that the response of the algorithm *A* would have to be divided by 10 to match the algorithm *B* response. The closer to 100% is the result of the function *gap*(*B, A*), better the algorithm *B* is compared to an algorithm *A*. The same measure applies to the run time of the algorithms. The order of the algorithms in the function entry is *gap*(*V NDB, MIP* ). The values of *gap*() *≥* 100% are due to rounding.

*z*(*A*)

A set of experiments was performed using the generated instances with *n ∈*

*{*100*,* 200*,* 400*,* 800*}* and considering *k ∈ {*3*,* 4*,* 5*,* 6*}*. Tables [A.1](#_bookmark30) and [A.2](#_bookmark31) present the computational experiments results, i.e., the runtime (in seconds) and the upper bound (*ub*), for each instance. These results are used to compute the *gap*() presented in Table [A.3](#_bookmark32). Table [A.3](#_bookmark32) may be observed for a descriptive analysis of the results. First, the VNDB runtime is always shorter since the *gapt* values are greater than 60%, indicating a runtime of at least 2*.*5 times smaller. VNDB finds the optimal solution for 9 instances in the *k* = 3 column when *gapub* = 100% is observed. Furthermore, the number of times that the *ub* provided by the VNDB algorithm exceeds the one provided by the mathematical model ([2](#_bookmark4)) solved via CPLEX solver, which corresponds to the number of positive *gapub* observed, is a total of 39 of the 80 instances tested. In more detail, VNDB presented to be best in 10 instances of column *k* = 3, 8 instances of column *k* = 4, 8 instances of column *k* = 5, and 13 instances of column *K* = 6.

Finally, a set of statistical tests using objective function values *falg* as samples is performed. The tests aim to answer whether it is possible to assert with 95% confidence if the objective function values *fMIP* , associated to the best solution attained by CPLEX solver when solving the mathematical model ([2](#_bookmark4)), are smaller than the *ub* obtained through VNDB, given by *fV NDB*. The hypothesis test used is:

*H*0 : *fMIP ≥ fV NDB H*1 : *fMIP < fV NDB*

(8)

Table [3](#_bookmark15) shows the p-values of the statistical tests using objective function values *falg* as samples. The hypothesis test rejects the null hypothesis only in instances with *k* = 4. In this case, it is possible to state that there is a difference between the algorithms and this difference is significant, i.e., the *ub* obtained through VNDB is inferior to objective function values *fMIP* associated to the mathematical model

([2](#_bookmark4)). In other cases, there is no difference between the results.

Table 3

Results of the paired t-test with 95% confidence using the objective function values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student test | K=3 | K=4 | K=5 | K=6 |
| p-values | 16*,* 66% | 3*,* 77% | 7*,* 04% | 46*,* 12% |

# Conclusion

This article presented an adapted VNDB matheuristic for solving MWNPP and a comparison with the solution of the mathematical model using the CPLEX solver. Results were obtained using randomly generated instances with uniformly distributed elements.

Computational experiment results show that VNDB never reaches the maximum time available for execution due to neighborhood exhaustion and a large number of solved subproblems with non-optimal status. Despite this, VNDB has statistically as good results as the same MIP running for a shorter time interval. The only exception is its performance in instances where *k* = 3. A drawback found in the computational experiments is that the CPLEX solver does not seem to set time in a deterministic way on a multi-core processor. This causes some problems to take up to 4500 seconds more in MIP execution.

As future work, a more elaborate parameter analysis can be applied to the proposed VNDB so that it can use all the available time for the experiment. Thus the comparison with the MIP problem can be more fair.

# References

1. Caserta, M. and S. Voß, *Matheuristics: Hybridizing metaheuristics and mathematical programming*, Metaheuristics: Intelligent Problem Solving, Springer, Berlin (2009), pp. 1–38.
2. Faria, A. F., S. R. de Souza and C. A. Silva, *Variable neighborhood descent applied to multi-way number* *partitioning problem*, Electronic Notes in Discrete Mathematics **66** (2018), pp. 103 – 110.
3. Fischetti, M. and A. Lodi, *Local branching*, Mathematical programming **98** (2003), pp. 23–47.
4. Gent, I. P. and T. Walsh, *Analysis of heuristics for number partitioning*, Computational Intelligence

**14** (1998), pp. 430–451.

1. Graham, R. L., *Bounds for certain multiprocessing anomalies*, The Bell System Technical Journal **XLV**

(1966), pp. 1563–1581.

1. 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).
2. 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.

1. Korf, R. E., *A complete anytime algorithm for number partitioning*, Artificial Intelligence **106** (1998),

pp. 181–203.

1. Korf, R. E., *Multi-way number partitioning.*, in: *IJCAI*, Citeseer, 2009, pp. 538–543.
2. 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)*, 2013.

1. 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.
2. Pedroso, J. P. and M. Kubo, *Heuristics and exact methods for number partitioning*, European Journal of Operational Research **202** (2010), pp. 73–81.
3. Schreiber, E. L., “Optimal Multi-Way Number Partitioning,” Ph.D. thesis, University of California Los Angeles (2014).
4. Schreiber, E. L. and R. E. Korf, *Cached iterative weakening for optimal multi-way number partitioning*, in: *Proceedings of the Twenty-Eighth Annual Conference on Artificial Intelligence (AAAI-14) Quebec City, Canada*, 2014.

# A Appendix

Table A.1 Mathematical Model [2](#_bookmark4) Results

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=3 | | k=4 | | k=5 | | k=6 | |
| n | Time | ub | Time | ub | Time | ub | Time | ub |
| 100 | 4063,50 | 49240 | 4513,94 | 249893 | 4498,00 | 455322 | 4495,52 | 1200561 |
| 100 | 3980,16 | 55843 | 4479,21 | 209860 | 4525,60 | 1605769 | 4523,85 | 5603669 |
| 100 | 4179,41 | 32015 | 4457,30 | 427940 | 4510,53 | 140990 | 4532,32 | 2602217 |
| 100 | 4120,93 | 52069 | 4455,24 | 93999 | 4558,21 | 1182564 | 4485,31 | 4725930 |
| 100 | 3905,11 | 6408,5 | 4475,06 | 80173,5 | 4542,53 | 1102740 | 4553,10 | 2951999 |
| 200 | 4371,95 | 13166,5 | 4519,97 | 135307 | 4557,98 | 1123962 | 4587,85 | 1590144 |
| 200 | 4381,11 | 4431 | 4537,66 | 65535 | 4542,90 | 739145 | 4666,52 | 4826459 |
| 200 | 4402,95 | 8819 | 4524,11 | 166066 | 4631,01 | 638049 | 4562,95 | 3675251 |
| 200 | 4367,34 | 19287 | 4448,33 | 214235 | 4578,41 | 1169098 | 4621,10 | 1855193 |
| 200 | 4390,39 | 45946 | 4504,69 | 159946 | 4479,97 | 1656139 | 4562,59 | 2524163 |
| 400 | 4375,50 | 16053,5 | 4612,23 | 253728 | 4691,69 | 194989 | 4753,60 | 888856 |
| 400 | 4532,59 | 52235 | 4563,96 | 358952 | 4600,64 | 1119839 | 4663,89 | 6310376 |
| 400 | 4531,44 | 4809 | 4611,12 | 32728 | 4707,68 | 1849077 | 4670,92 | 3613928 |
| 400 | 4553,46 | 24003 | 4591,07 | 292372 | 4672,92 | 406028 | 4690,89 | 791927 |
| 400 | 4523,91 | 22076 | 4576,98 | 129955 | 4672,51 | 732207 | 4705,01 | 2657192 |
| 800 | 4805,05 | 2348 | 4846,90 | 121657 | 4853,19 | 490825 | 4932,21 | 1054196 |
| 800 | 4674,52 | 69290 | 4877,63 | 63272 | 4874,87 | 363647 | 4894,91 | 4864688,999023 |
| 800 | 4782,10 | 19733,5 | 4851,47 | 436698 | 4859,28 | 646373 | 4896,03 | 1937925 |
| 800 | 4728,13 | 7854 | 4829,16 | 97683 | 4884,95 | 515063 | 4834,07 | 2996054,999023 |
| 800 | 4792,19 | 30999 | 4844,46 | 116701 | 4851,21 | 533006 | 4921,96 | 1044665 |

All resources used for experiments of this paper (including codes, in- stances, and results) are available in [https://github.com/AlexandreFrias/](https://github.com/AlexandreFrias/AlgorithmsLAGOS2019) [AlgorithmsLAGOS2019](https://github.com/AlexandreFrias/AlgorithmsLAGOS2019).

Table A.2 VNDB Results

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=3 | | k=4 | | k=5 | | k=6 | |
| n | Time | ub | Time | ub | Time | ub | Time | ub |
| 100 | 912,14 | 39322 | 744,14 | 458052 | 680,49 | 596504 | 658,14 | 596504 |
| 100 | 1558,98 | 0 | 985,70 | 233894 | 905,41 | 596225 | 417,39 | 2996624 |
| 100 | 878,13 | 43515 | 667,90 | 295726 | 552,37 | 1132949 | 893,61 | 1379484 |
| 100 | 711,36 | 0 | 718,82 | 585619 | 469,65 | 468677 | 867,56 | 1848561 |
| 100 | 582,89 | 0 | 1079,56 | 229865 | 875,49 | 1425028 | 531,09 | 3014678 |
| 200 | 1490,28 | 0 | 1233,08 | 128692 | 1210,58 | 689690 | 1204,65 | 681721 |
| 200 | 1000,42 | 33602 | 1256,11 | 59067,5 | 800,97 | 1969700 | 823,28 | 2017573 |
| 200 | 1094,12 | 86025 | 1252,33 | 337142 | 847,00 | 642697 | 831,02 | 1674668 |
| 200 | 1448,89 | 46660 | 876,20 | 421421 | 838,17 | 821353 | 579,58 | 4273737 |
| 200 | 1441,05 | 76803 | 1135,23 | 216298 | 1231,48 | 747272 | 427,39 | 4989050 |
| 400 | 1744,63 | 0 | 1732,56 | 192672 | 697,12 | 1013347 | 598,29 | 560329 |
| 400 | 1000,71 | 0 | 1100,20 | 244158 | 778,44 | 877283 | 473,71 | 2102136 |
| 400 | 1457,58 | 0 | 636,83 | 103158 | 742,47 | 134151 | 529,31 | 1154835 |
| 400 | 1014,13 | 73047 | 1027,87 | 72002 | 1065,58 | 795390 | 557,70 | 740184 |
| 400 | 615,47 | 0 | 628,12 | 153911 | 399,66 | 930850 | 546,27 | 1298980 |
| 800 | 1534,66 | 66186 | 807,00 | 60500 | 1224,68 | 2681189 | 1148,09 | 3373820 |
| 800 | 695,73 | 0 | 968,98 | 2234417 | 1087,70 | 1456731 | 1043,90 | 3383655 |
| 800 | 670,97 | 91676 | 892,95 | 377468 | 1113,92 | 586678 | 1138,06 | 9116751 |
| 800 | 1179,86 | 105889 | 792,00 | 1757558 | 1183,25 | 7326466 | 1128,57 | 8909755 |
| 800 | 1407,38 | 91542 | 812,39 | 618097 | 972,31 | 9596348 | 1121,96 | 4947780 |

Table A.3

Comparison between VNDB and Mathematical Model [2](#_bookmark4) using *gap*(*vndb, mip*) on upper bound and runtime.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=3 | | k=4 | | k=5 | | k=6 | |
| n | *gapt* | *gapub* | *gapt* | *gapub* | *gapt* | *gapub* | *gapt* | *gapub* |
| 100 | 77,55% | 20,14% | 83,51% | -83,30% | 84,87% | -31,01% | 85,36% | 50,31% |
| 100 | 60,83% | 100,00% | 77,99% | -11,45% | 79,99% | 62,87% | 90,77% | 46,52% |
| 100 | 78,99% | -35,92% | 85,02% | 30,90% | 87,75% | -703,57% | 80,28% | 46,99% |
| 100 | 82,74% | 100,00% | 83,87% | -523,01% | 89,70% | 60,37% | 80,66% | 60,88% |
| 100 | 85,07% | 100,00% | 75,88% | -186,71% | 80,73% | -29,23% | 88,34% | -2,12% |
| 200 | 65,91% | 100,00% | 72,72% | 4,89% | 73,44% | 38,64% | 73,74% | 57,13% |
| 200 | 77,17% | -658,33% | 72,32% | 9,87% | 82,37% | -166,48% | 82,36% | 58,20% |
| 200 | 75,15% | -875,45% | 72,32% | -103,02% | 81,71% | -0,73% | 81,79% | 54,43% |
| 200 | 66,82% | -141,93% | 80,30% | -96,71% | 81,69% | 29,74% | 87,46% | -130,37% |
| 200 | 67,18% | -67,16% | 74,80% | -35,23% | 72,51% | 54,88% | 90,63% | -97,65% |
| 400 | 60,13% | 100,00% | 62,44% | 24,06% | 85,14% | -419,69% | 87,41% | 36,96% |
| 400 | 77,92% | 100,00% | 75,89% | 31,98% | 83,08% | 21,66% | 89,84% | 66,69% |
| 400 | 67,83% | 100,00% | 86,19% | -215,20% | 84,23% | 92,74% | 88,67% | 68,04% |
| 400 | 77,73% | -204,32% | 77,61% | 75,37% | 77,20% | -95,90% | 88,11% | 6,53% |
| 400 | 86,40% | 100,00% | 86,28% | -18,43% | 91,45% | -27,13% | 88,39% | 51,11% |
| 800 | 68,06% | -2718,82% | 83,35% | 50,27% | 74,77% | -446,26% | 76,72% | -220,04% |
| 800 | 85,12% | 100,00% | 80,13% | -3431,45% | 77,69% | -300,59% | 78,67% | 30,44% |
| 800 | 85,97% | -364,57% | 81,59% | 13,56% | 77,08% | 9,24% | 76,76% | -370,44% |
| 800 | 75,05% | -1248,22% | 83,60% | -1699,25% | 75,78% | -1322,44% | 76,65% | -197,38% |
| 800 | 70,63% | -195,31% | 83,23% | -429,64% | 79,96% | -1700,42% | 77,21% | -373,62% |