Computers and Mathematics with Applications 60 (2010) 2302–2308

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Integer linear programming model for multidimensional two-way number partitioning problem

Jelena Kojić

University of Belgrade, Faculty of Mathematics, Studentski trg 16, 11000 Belgrade, Serbia

ARTICLE INFO

Article history:
Received 27 July 2009
Received in revised form 9 August 2010
Accepted 9 August 2010

Keywords:
Number partitioning
Integer linear programming
Combinatorial optimization

ABSTRACT

This paper introduces a multidimensional generalization of the two-way number partitioning problem, as well as an integer linear programming formulation of the problem. There are n binary variables and $2p$ constraints. The numerical experiments are made on a randomly generated set. In view of its integer linear programming formulation, tests are run in CPLEX. This NP-hard problem uses a set of vectors rather than a set of numbers. The presented experimental results indicate that the generalized problem is much harder than the initial problem.

1. Introduction

The two-way number partitioning problem (TWNPP) calls for splitting an original set into two subsets so that their sums should be equal or approximately equal. Various methods have been developed for resolving this problem, such as: total enumeration or brute force, the Horowitz–Sahni algorithm, the greedy heuristic, the Karmarkar–Karp heuristic, and the complete anytime algorithm. Pedroso gives some methods for solving this problem. More methods for solving the TWNPP can be seen in [1].

In the case of total enumeration, the sum of every possible subset is calculated and the subset returned with the sum closest to one-half of the sum total of all the elements of the original set. If n is the number of elements in the original set, then the time complexity of this algorithm is $O(2^n)$. This algorithm is impractical for instances of greater dimensions.

The Horowitz–Sahni algorithm [2] is a slight improvement on total enumeration in terms of time needed for its execution, since its time complexity is $O(2^{\frac{n}{2}})$. The original set (with an n number of elements) is split into two subsets with $\frac{n}{2}$ elements each. If n is an odd number, the relevant subsets will each have $[\frac{n}{2}]$ and $[\frac{n}{2}]+1$ elements ($[x]$ denotes a whole part of x). The sums of elements of all their subsets are calculated and memorized for each of the subsets. Two new subsets are generated from the memorized sums. This algorithm requires extensive memory, so that it, too, is impractical in instances of greater dimensions.

The greedy heuristic for this problem starts by sorting the numbers of the original set in a non-growing sequence. The largest element is placed in the first subset and each following element is placed in a subset with a lower sum. As is the case of other greedy heuristics, this algorithm is very fast (its time complexity is $O(n \cdot \log(n))$), but the results obtained are usually far from optimal.

The Karmarkar–Karp heuristic [3], too, first sorts the numbers in the original set in a non-growing sequence. The two biggest elements are placed in different subsets and, instead of them, their difference is observed, placed in the original sorted set, and treated as a new element of the set. The algorithm again eliminates the two largest elements and places their difference in the original subset, all the way until there is only one element left in the set, which is the difference between two subsets. This algorithm, too, usually produces far from optimal results, so that it is added upon until an exact method is reached: the complete anytime algorithm (called Complete Karmarkar–Karp algorithm) [4]. The time complexity of the Karmarkar–Karp heuristic is $O(n \cdot \log(n))$.

E-mail address: k_jelena@yubc.net.

0898-1221/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2010.08.024

Pedroso's idea [5] is drawn from branch-and-bound, breadth first search, and beam search. In each depth of the search tree, the nodes are sorted according to the number of times that a sum appears in the branch-and-bound tree until the node is reached.

The multidimensional two-way number partitioning problem (MDTWNPP) is a generalization of the TWNPP where a set of vectors is partitioned rather than a set of numbers. Instead of one sum there is a sum per every coordinate and those sums should be exactly or approximately equal. The TWNPP is a special case of the MDTWNPP, where the dimension of the problem is 1. Since the TWNPP is an NP-hard problem, then the MDTWNPP as its generalization is also an NP-hard problem, see [4].

An important characteristic of the two-way number partitioning problem is that its computational complexity depends on the type of input numbers, i.e. number precision, as can be seen from [6,7]. If there are more numbers the problem is easier, so its complexity must be preserved with a larger number of digits.

This is not the case with the multidimensional two-way number partitioning problem, because its complexity does not decrease even with a large number of inputs. Moreover, MDTWNPP is very hard to solve even with a moderate number of digits, as the experimental results show.

The multidimensional two-way number partitioning problem is similar to clustering. Clustering is an unsupervised pattern classification technique that is defined as a group of a particular number of objects divided into a certain number of clusters. The number of partitions/clusters may or may not be known in advance. Detailed presentation of the work on clustering is out of this paper's scope, but some of the new and successful ones are [8–10].

Although the TWNPP and the clustering problems are similar, they are quite different in nature. In clustering, the objective is to group objects into clusters so that, in some metrics, the overall sum (or maximum) of distances between elements in the same cluster is minimized. On the other hand, the MDTWNPP does not deal with any distances between elements, but just minimizes the overall sums of elements (for every coordinate) in the sets. In this case, the metric is not necessary, because here the far elements can be in the same set, with the same chances as the near elements.

None of the said methods can be applied directly to solving the TWNPP.

Total enumeration is applicable, but its complexity rises exponentially with the increase of the dimension, so that it produces no result. There were 2^{50} operations, or approximately 10^{15}, which is intractable for computing.

It would be very difficult to apply the Horowitz–Sahni algorithm, because it only has one sum, whereas the MDTWNP has more than one different sum for every dimension.

Greedy heuristics cannot be applied because of the sorting of numbers, since one dimension here can have different sorting than another dimension. Since they are sorted according to different dimensions, different results can be obtained.

The Karmarkar–Karp heuristic cannot be applied, either, because of the sorting of elements and, by corollary, its extension to the complete anytime algorithm is inapplicable, too.

The Pedroso Algorithm also includes sorting (in the branch part), but with weights. In the case of the MDTWNPP, there would be a d number of detractions, which would yield not just one value, but a min and a max for every dimension, which would produce a range. If we changed the minimum for one dimension, the question is how that would affect the other dimensions, that is, the correction of one dimension could affect another. It would be very difficult to create a good criterion for sorting. Perhaps the Pedroso Algorithm could be applied in theory, but not directly. It would have to be amended, but there is a good chance that, in that case, it would lose its efficacy. This can be concluded by taking a look at the experimental results, because the important thing about the TWNPP is the number of decimal points, whereas the result for the MDTWNPP does not show that an increase in decimal points changes anything.

The two-way number partitioning problem is applicable in cryptography, time-scheduling, and problems with non-identical I/O capacities that appear in database processing. For more information about the practical application of this problem, see [11]. Further uses of this problem can be seen in [12–14]. A statistical approach to the TWNPP is given in [15].

Example: There are two identical computers, a string of tasks, and time necessary for completing a task at a given computer, but tasks cannot be shared. Each task is assigned to one of the computers and the job is to complete the tasks in the shortest possible time period. In other words, the time necessary for carrying out the tasks in the two sets is divided in such a way that the sum totals of the times in each subset are roughly equal.

A problem that is closely linked to this one is the problem of the sum of a subset. There is a given set of numbers and a constant c and the idea is to find a subset of the original set in such a way that the sum of the elements in the subset should be exactly c. This problem boils down to a problem of a two-way partitioning of numbers in the following way. Let s be the sum of elements in the original set; if $c < \frac{s}{2}$, we take c to be $s - c$. We introduce a new element in the set, d, $c = \frac{s+d}{2}$, i.e., $d = 2c - s$. If the sums obtained in the subsets are equal, the sum in the subset that does not contain d will be exactly c. If the optimum solution of the two-way number partitioning problem in the new set is greater than zero, the original problem does not have a solution, i.e., there is no subset with the sum c.

Some of the practical applications of the MDTWNPP (where the dimensions are greater than 1) are as follows:

- Two travel agents need to have as closely similar revenues from package tours as possible; the parameters under consideration are the price of the tour, the mileage (due to petrol consumption), special offers (such as restaurants, shopping tours, visits to museums, and so on). This is a three-dimensional, two-way number partitioning problem.
- Two police patrols need to be assigned tasks in such a way as to have the labor divided as closely between them as possible in terms of mileage, degree of risk (considering criminal activity on their respective beats), and time needed for completing their beats.

2. Integer linear programming formulation

Discrete optimization problems should be represented as integer programming problems in order to apply different well-known optimization techniques in solving them [16–19]. Pursuant to this idea, what is given here is an integer linear programming (ILP) formulation for the MDTWNPP, so that any ILP solver can be used for solving this problem.

Let S be a set whose elements are p-tuple of real numbers and $n = |S|$. The elements should be divided into two sets, S_1 and S_2, so that $S_1 \cup S_2 = S, S_1 \cap S_2 = \emptyset$ holds true and the sums of the elements in the subsets S_1 and S_2 are equal or almost equal for all coordinates.

Let w_{ij} be the j-th coordinate of the i-th element $w_i = (w_{i1}, \ldots, w_{ip})$ in the set S, and

$$s_j = \sum_{i=1}^{n} w_{ij}, \quad j = 1, \ldots, p. \tag{1}$$

If the variable is

$$x_i = \begin{cases} 1, & i \in S_1 \\ 0, & i \in S_2 \end{cases} \tag{2}$$

and the variable t denotes the greatest difference in sums per coordinate, i.e.,

$$t = \max \left| \sum_{i \in S_1} w_{ij} - \sum_{i \in S_2} w_{ij} \right|, \tag{3}$$

the mathematical expression of the ILP formulation is:

$$\min t \tag{4}$$

subject to:

$$-0.5 \cdot t + \sum_{i=1}^{n} w_{ij} \cdot x_i \leq 0.5 \cdot s_j, \quad j = 1, \ldots, p \tag{5}$$

$$0.5 \cdot t + \sum_{i=1}^{n} w_{ij} \cdot x_i \geq 0.5 \cdot s_j, \quad j = 1, \ldots, p \tag{6}$$

$$x_j \in \{0, 1\}, \quad i = 1, \ldots, n. \tag{7}$$

Condition (4) defines whether the sum of elements in subsets S_1 and S_2 is equal to $(\min t = 0)$ or as close to zero as possible. In view of the fact that (3) is not linear, because it contains an absolute value, it is equivalent with conditions (5) and (6) when we shed the absolute value and use the definition of the variables x_i.

As can be seen, there are n binary variables and $2 \cdot p$ constraints.

From constraint (3) it follows:

$$\max_j \left| \sum_{i \in S_1} w_{ij} - \sum_{i \in S_2} w_{ij} \right| = t \tag{8}$$

$$\forall j \quad \left| \sum_{i \in S_1} w_{ij} - \sum_{i \in S_2} w_{ij} \right| \leq t \tag{9}$$

$$-t \leq \sum_{i \in S_1} w_{ij} - \sum_{i \in S_2} w_{ij} \leq t \tag{10}$$

the sum of the elements in sets S_1 and S_2 can be written as follows:

$$\sum_{i \in S_1} w_{ij} = \sum_{i} w_{ij} \cdot x_i \tag{11}$$

$$\sum_{i \in S_2} w_{ij} = \sum_{i} w_{ij} \cdot (1 - x_i). \tag{12}$$

By inserting the equalities (11) and (12) into the inequality (10), we get:

$$-t \leq \sum_{i} w_{ij} \cdot x_i - \sum_{i} w_{ij} + \sum_{i} w_{ij} \cdot x_i \leq t \tag{13}$$

$$-t \leq -s_j + 2 \cdot \sum_{i} w_{ij} \cdot x_i \leq t \tag{14}$$

from which the required constraints are derived:

Table 1
Results of CPLEX.

Inst	$Best_{sol}$	t (s)	*Iter*	*Inst*	$Best_{sol}$	t (s)	*Iter*
50_2a	3.204	552.60	19 268 295	100_2a	19.513	482.98	13 263 420
50_2b	9.193	63.75	2 093 979	100_2b	3.915	266.72	7 274 748
50_2c	9.191	161.90	5 502 837	100_2c	7.975	836.79	22 525 357
50_2d	4.753	92.01	3 076 040	100_2d	3.055	766.01	20 309 753
50_2e	6.719	1340.84	39 654 688	100_2e	2.077	1399.84	36 045 821
50_3a	303.581	465.91	23 213 799	100_3a	130.255	975.09	35 974 963
50_3b	350.828	430.88	22 540 402	100_3b	97.935	80.71	2 996 108
50_3c	152.089	774.38	35 328 618	100_3c	263.199	383.98	13 876 910
50_3d	102.516	1378.31	70 842 460	100_3d	247.043	1523.05	59 522 440
50_3e	217.903	312.16	17 704 635	100_3e	153.615	462.47	16 018 141
50_4a	909.765	1663.43	130 662 143	100_4a	520.496	664.68	30 900 280
50_4b	1 272.224	1581.08	99 028 578	100_4b	1 021.628	835.00	36 856 779
50_4c	461.161	1275.95	74 795 550	100_4c	908.240	232.91	11 097 609
50_4d	1 024.681	827.70	59 886 756	100_4d	1 096.223	1678.59	73 112 661
50_4e	1 199.574	498.30	30 889 244	100_4e	517.443	102.99	5 806 269
50_5a	926.164	281.77	20 989 584	100_5a	2 441.489	570.23	28 305 729
50_5b	3 202.875	1192.94	92 896 763	100_5b	2 825.848	284.94	15 018 616
50_5c	2 696.703	143.42	10 152 689	100_5c	2 833.222	1236.07	65 032 263
50_5d	2 275.792	357.38	30 559 828	100_5d	2 975.937	15.03	784 370
50_5e	4 823.935	197.43	13 955 412	100_5e	4 160.207	1072.36	55 979 494
50_10a	16 176.578	1432.87	104 250 858	100_10a	17 699.079	46.88	3 816 695
50_10b	19 560.318	5.97	436 872	100_10b	18 993.443	704.88	51 700 324
50_10c	17 757.097	1308.68	96 714 821	100_10c	15 386.568	703.95	51 926 186
50_10d	14 925.023	1104.70	89 495 364	100_10d	18 276.246	337.87	24 748 211
50_10e	15 369.009	472.44	33 987 445	100_10e	16 516.277	689.61	51 541 610
50_15a	33 208.019	1777.83	104 075 782	100_15a	32 143.237	1082.82	63 794 024
50_15b	35 003.301	1121.87	66 660 887	100_15b	28 723.793	202.41	13 081 002
50_15c	29 920.923	1443.76	92 815 895	100_15c	33 363.206	1157.74	69 797 033
50_15d	21 652.841	1594.98	103 362 807	100_15d	30 706.170	1098.49	63 825 249
50_15e	31 800.690	332.25	20 719 179	100_15e	30 253.623	280.81	18 389 733
50_20a	52 826.340	71.83	3 773 182	100_20a	49 992.607	1643.40	78 335 212
50_20b	51 917.902	1378.83	59 626 377	100_20b	46 691.489	1518.05	74 429 911
50_20c	50 560.864	493.27	23 955 706	100_20c	45 739.714	1042.18	49 910 424
50_20d	53 955.965	166.26	8 813 405	100_20d	45 371.992	406.33	21 576 175
50_20e	48 281.499	234.75	11 596 103	100_20e	52 315.704	1271.84	57 337 406

$$-\frac{t}{2} \leq -\frac{s_j}{2} + \sum_i w_{ij} \cdot x_i \leq \frac{t}{2}. \tag{15}$$

From this follow constraints (5) and (6).

Example 2.1. $S = \{(1\ \ 3), (5\ \ 5), (3\ \ -2), (-3\ \ 12)\}$ For S_1 and S_2 we have several candidates:

- $S_1 = \{(1\ \ 3)\}, S_2 = \{(5\ \ 5), (3\ \ -2), (-3\ \ 12)\}$, sums are: $(1\ \ \ 3)$, $(5\ \ \ 15)$, and the difference is $(4\ \ \ 12)$, $t = 12$ (max = 12)
- $S_1 = \{(5\ \ 5)\}, S_2 = \{(1\ \ 3), (3\ \ -2), (-3\ \ 12)\}$, sums are: $(5\ \ 5)$, $(1\ \ 13)$, and the difference is $(4\ \ 8)$, $t = 8$
- $S_1 = \{(3\ \ -2)\}, S_2 = \{(1\ \ 3), (5\ \ 5), (-3\ \ 12)\}$, sums are: $(3\ \ -2)$, $(3\ \ 20)$, and the difference is $(0\ \ 22)$, $t = 22$
- $S_1 = \{(-3\ \ 12)\}, S_2 = \{(1\ \ 3), (5\ \ 5), (3\ \ -2)\}$, sums are: $(-3\ \ 12)$, $(9\ \ 6)$, and the difference is $(12\ \ 6)$, $t = 12$
- $S_1 = \{(1\ \ 3), (5\ \ 5)\}, S_2 = \{(3\ \ -2), (-3\ \ 12)\}$, sums are: $(6\ \ 8)$, $(0\ \ 10)$, and the difference is $(6\ \ 2)$, $t = 6$
- $S_1 = \{(1\ \ 3), (3\ \ -2)\}, S_2 = \{(5\ \ 5), (-3\ \ 12)\}$, sums are: $(4\ \ 1)$, $(2\ \ 17)$, and the difference is $(2\ \ 16)$, $t = 16$
- $S_1 = \{(1\ \ 3), (-3\ \ 12)\}, S_2 = \{(5\ \ 5), (3\ \ -2)\}$, sums are: $(-2\ \ 15)$, $(8\ \ 3)$, and the difference is $(10\ \ 12)$, $t = 12$.

The minimum of the maximal elements is in the fifth case. It means that the optimal solution is: $S_1 = \{(1\ \ 3), (5\ \ 5)\}$, $S_2 = \{(3\ \ -2), (-3\ \ 12)\}$, $\min t = 6$.

3. Computational results

The tests were run on an Intel 2.5 GHz with 4 GB RAM memory, in CPLEX 10.1. The instances 500_20 were generated randomly, where 500 represents the number of elements and 20, the dimension. There were five 500_20 instances (a, b, c, d, e). Thus, all other instances were in the nature of subsets of the original instance, that is, sub-matrixes where the first element of a sub-matrix was the first element of the 500_20 instance. The tests were run for a maximum of 30 min each.

The first and fifth columns in Tables 1, 2 and 3 give the names of the instances, and the second and sixth, the best results obtained. Since the CPLEX did not finish its work for any instance of the multidimensional case, within the set running

Table 2
Results of CPLEX.

Inst	$Best_{sol}$	*t* (s)	*Iter*	*Inst*	$Best_{sol}$	*t* (s)	*Iter*
200_2a	11.463	766.53	16 188 931	300_2a	2.744	1753.38	28 427 389
200_2b	3.919	915.91	17 800 795	300_2b	6.958	682.50	10 704 468
200_2c	0.000	704.92	14 478 841	300_2c	2.730	1273.13	21 331 258
200_2d	2.691	1146.03	24 107 359	300_2d	0.881	1446.18	23 339 152
200_2e	0.971	661.39	13 244 769	300_2e	1.522	1647.92	30 738 744
200_3a	181.368	1773.64	46 504 721	300_3a	6.100	941.22	20 952 772
200_3b	137.584	256.78	7 895 891	300_3b	110.139	1097.10	26 442 330
200_3c	3.059	537.67	15 797 017	300_3c	226.933	459.49	11 641 494
200_3d	224.645	1129.01	33 408 581	300_3d	137.587	952.38	20 544 786
200_3e	120.542	844.37	24 316 786	300_3e	188.581	263.35	5 398 430
200_4a	537.018	372.95	12 851 010	300_4a	15.268	537.98	17 237 858
200_4b	1 188.248	32.82	1 175 739	300_4b	1 068.095	293.00	7 707 402
200_4c	6.109	469.97	19 311 854	300_4c	900.620	355.95	10 162 283
200_4d	1 094.743	92.75	3 241 458	300_4d	1 004.401	972.48	27 330 662
200_4e	1 264.715	24.06	755 091	300_4e	908.869	449.44	12 282 125
200_5a	1 931.064	440.70	13 629 765	300_5a	1 847.760	320.52	8 103 603
200_5b	2 734.271	188.68	6 130 206	300_5b	4 195.209	138.36	2 740 278
200_5c	3 576.930	236.02	8 110 667	300_5c	2 658.010	1621.06	7 048 573
200_5d	2 782.748	58.78	1 921 413	300_5d	2 396.939	1339.89	28 668 967
200_5e	3 798.611	98.85	3 641 988	300_5e	2 499.651	135.62	3 392 558
200_10a	16 530.321	146.75	6 114 388	300_10a	16 112.376	40.06	1 345 319
200_10b	19 616.619	151.34	5 772 542	300_10b	1 9954.971	205.00	4 131 933
200_10c	16 158.656	759.02	21 734 005	300_10c	15 996.203	184.77	4 575 659
200_10d	17 399.449	1640.67	47 068 191	300_10d	20 282.178	1768.84	6 799 602
200_10e	18 107.353	151.13	5 383 595	300_10e	19 620.941	70.30	2 174 884
200_15a	35 139.957	669.95	38 199 816	300_15a	37 524.309	1678.59	14 001 901
200_15b	34 575.029	649.44	37 541 113	300_15b	34 673.445	737.63	10 252 774
200_15c	35 016.095	934.43	51 772 908	300_15c	30 553.455	208.90	6 455 131
200_15d	33 160.395	742.52	41 007 121	300_15d	36 264.630	179.13	6 269 399
200_15e	29 600.126	493.50	27 866 023	300_15e	32 237.793	186.39	6 586 528
200_20a	44 991.718	872.96	33 421 574	300_20a	47 297.493	1302.81	43 977 618
200_20b	49 884.338	377.89	14 734 151	300_20b	44 127.831	940.69	30 756 243
200_20c	48 451.593	334.35	13 024 858	300_20c	43 594.894	1033.94	26 559 816
200_20d	43 631.462	382.39	14 971 227	300_20d	48 814.817	1338.33	38 068 102
200_20e	41 768.116	1247.70	46 103 313	300_20e	50 067.495	799.10	26 721 473

time 30 min, it has been omitted from the tables. On the other hand, CPLEX solved all instances in the single dimension case very quickly, as expected from the theoretical analysis described in [6,7]. Since the one-dimensional problem is always optimally solved in less than 1 s, these results have also been omitted from the tables. Since the execution time was limited to 30 min, it is significant to note after how long the best solution was attained, so that the third and seventh columns show the approximate time in seconds in which the best solution was reached. The number of iterations in which the best result was achieved first is shown in the fourth and eight columns.

Gaps are not listed in the tables, because they could not be accurately evaluated since the dual was less than 0.01. The primal was dropping, but the dual was not rising, so that execution time was reduced to 30 min, otherwise it would have been extremely long. The gap was large, almost 100%. From the experimental results it is clear that this problem is very difficult to solve. Obviously, the constraints in the ILP formulation are natural, but not tight.

Experimental results show that ideal division was achieved in only one instance (where the difference in the sums for all coordinates was 0); for the instance 200_2c, this was the optimum solution. For the other instances, the optimum solution was not verified, because CPLEX was run for 1800 s.

Solutions ranged up to 54 921.900—the highest value was attained for the instance 500_2b. The shortest time in which the best solution was attained was 5.97 s for the instance 50_10b, while the longest time was 1797.03 s for 500_2b. The number of iterations for the best result ranged from 285 886 (for 500_5b) to 130 662 143 (for 50_4a). The result shows that instances with the minimum number of iterations from each group (50,100, 200, 300, 400, 500) coincide with instances with the shortest execution time. This is not the case with the maximum number of iterations, because instances with the highest number of iterations coincide with instances with the longest execution time only in the groups with the element numbers 50 and 400—this includes the highest number of iterations for all instances, 130 662 143 for instance 50_4a, with the time for attaining the best result 1663.43 s. When one looks at the shortest times for the groups of instances 50, 100, 200, 300, 400, and 500, in which the best results were attained, one can see that, for the group 400, the fastest attained best result was for the instance of the dimension 15 (400_15e), while on the other hand, looking at the longest times when the best result was attained, there were also instances with only two dimensions, such as 500_2a with the time 1797.03, and also with only 50 elements, for example 50_4a, with the time 1663.43 s. This may indicate that this problem (MDTWNPP)

Table 3
Results of CPLEX.

Inst	$Best_{sol}$	*t* (s)	*Iter*	*Inst*	$Best_{sol}$	*t* (s)	*Iter*
400_2a	12.592	1708.50	24 249 712	500_2a	0.620	1745.39	19 756 812
400_2b	1.530	830.32	11 822 492	500_2b	1.879	1797.03	21 794 456
400_2c	4.354	956.34	13 490 499	500_2c	2.013	1076.04	12 589 787
400_2d	4.420	151.52	2 268 427	500_2d	1.003	1687.53	20 466 339
400_2e	5.185	1427.76	20 443 716	500_2e	0.913	1726.60	19 387 553
400_3a	194.372	704.74	13 888 100	500_3a	213.539	1698.09	28 897 868
400_3b	175.900	478.81	10 375 076	500_3b	142.160	806.88	14 344 521
400_3c	220.830	735.10	14 618 003	500_3c	270.729	466.64	7 088 168
400_3d	257.849	80.78	1 512 966	500_3d	13.736	1134.38	20 265 591
400_3e	194.681	176.90	3 052 918	500_3e	7.625	1091.25	19 191 606
400_4a	1 409.159	518.64	10 133 909	500_4a	1 562.920	171.29	2 251 129
400_4b	749.984	1237.82	21 076 835	500_4b	1 186.722	429.41	7 638 160
400_4c	914.349	200.49	3 921 731	500_4c	1 093.756	63.74	1 056 574
400_4d	941.826	446.63	9 041 071	500_4d	4.580	714.37	12 619 722
400_4e	902.761	576.52	12 945 268	500_4e	1 410.694	106.31	1 155 999
400_5a	3 737.767	79.90	1 082 710	500_5a	3 665.484	36.99	510 718
400_5b	1 494.313	1225.90	2 801 474	500_5b	4 448.741	21.29	285 886
400_5c	2 680.899	53.76	880 065	500_5c	3 511.837	523.39	1 686 018
400_5d	2 539.113	135.63	2 285 917	500_5d	2 597.644	81.42	1 158 097
400_5e	1 634.702	65.29	11 930 543	500_5e	2 572.429	955.58	2 334 203
400_10a	14 836.579	1622.34	6 174 237	500_10a	19 183.301	1718.29	6 126 005
400_10b	17 918.141	1215.03	5 732 163	500_10b	12 161.350	128.48	2 314 014
400_10c	21 213.818	1703.88	6 665 466	500_10c	16 594.760	368.13	3 474 077
400_10d	15 212.906	1283.81	6 083 804	500_10d	20 284.381	1699.01	6 908 845
400_10e	16 369.531	1530.48	7 752 182	500_10e	15 548.670	1680.47	8 534 405
400_15a	37 574.022	926.03	11 017 145	500_15a	30 316.775	1055.81	8 993 017
400_15b	34 390.093	62.52	2 015 214	500_15b	31 878.383	1591.08	13 619 223
400_15c	37 161.817	1463.43	13 811 091	500_15c	32 792.472	803.77	8 245 079
400_15d	30 019.198	1203.22	13 561 881	500_15d	35 555.260	881.27	9 182 384
400_15e	32 561.093	26.19	745 525	500_15e	30 806.719	455.06	6 868 706
400_20a	41 974.284	767.30	12 968 207	500_20a	48 281.977	1000.12	16 937 307
400_20b	46 751.348	354.05	9 031 221	500_20b	54 921.900	237.63	6 289 735
400_20c	47 259.514	313.95	6 764 285	500_20c	41 578.884	1382.98	19 133 997
400_20d	51 544.421	31.38	880 003	500_20d	54 293.200	1728.58	18 030 728
400_20e	48 792.272	251.36	7 092 642	500_20e	41 092.622	1713.03	19 572 601

is more difficult than the TWNPP, because, for example, for the group with 300 elements each, both the longest and the shortest times for attaining the best solution are for 10 dimensions—for 300_10d is the longest: 1768.84 and for 300_10a is the shortest: 40.06, while on the other hand, an extremely long time for attaining the best result is also for 2 dim, that is, 300_2a with the time 1753.38.

4. Conclusion

This paper is devoted to the multidimensional two-way number partitioning problem. It introduces the integer linear programming formulation and proves the correctness of the corresponding formulation. The numbers of variables and constraints were relatively small compared to the dimension of the problem.

Numerical experiments were performed using randomly generated instances. Numerical results obtained by CPLEX based on this ILP formulation suggest that this generalization is much more complex than the base problem.

The merit of the work is that it has produced a model that should enable the use of different well-known optimization techniques in solving the MDTWNPP.

Further work can be directed toward designing exact methods using the proposed ILP formulation, implementing some metaheuristic, and solving similar problems.

References

[1] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation: part II. Graph coloring and number partitioning, Operational Research 39 (3) (1991) 378–406.
[2] E. Horowitz, S. Sahni, Computing partitions with applications to the Knapsack problem, Journal of ACM 21 (2) (1974) 277–292.
[3] N. Karmarkar, R.M. Karp, The differencing method of set partitioning, Technical Report UCB/CSD 82/l 13, Computer Science Division, University of California, Berkeley, CA, 1982.
[4] R.E. Korf, A complete anytime algorithm for number partitioning, Artificial Intelligence 106 (1998) 181–203.
[5] J.P. Pedroso, M. Kubo, Heuristics and exact methods for number partitioning, European Journal of Operational Research 202 (1) (2010) 73–81.

[6] S. Mertens, Phase transition in the number partitioning problem, Physical Review Letters 81 (20) (1998) 4281–4284.
[7] S. Mertens, The easiest hard problem: number partitioning, in: A. Percus, G. Istrate, C. Moore (Eds.), Computational Complexity and Statistical Physics, Oxford University Press, New York, 2006, pp. 125–139.
[8] W. Song, S.C. Park, Genetic algorithm for text clustering based on latent semantic indexing, Computers and Mathematics with Applications 57 (11–12) (2009) 1901–1907.
[9] E. Ficarra, G. De Micheli, S. Yoon, L. Benini, E. Macii, Joint co-clustering: co-clustering of genomic and clinical bioimaging data, Computers and Mathematics with Applications 55 (5) (2008) 938–949.
[10] M.S. Yang, D.C. Lin, On similarity and inclusion measures between type-2 fuzzy sets with an application to clustering, Computers and Mathematics with Applications 57 (6) (2009) 896–907.
[11] M. Koyuturk, Hypergraph based declustering for multi-disk databases, A Thesis for the Degree of Master of Science, 2000, pp. 50–51.
[12] B. Alidaee, F. Glover, G.A. Kochenberger, C. Rego, A new modeling and solution approach for the number partitioning problem, Journal of Applied Mathematics and Decision Sciences 9 (2) (2005) 113–121.
[13] H. Bauke, S. Franz, S. Mertens, Number partitioning as a random energy model, Journal of Statistical Mechanics: Theory and Experiment 04 (2004) P04003. http://stacks.iop.org/1742-5468/2004/P04003.
[14] H. De Raedt, K. Michielsen, K. De Raedt, S. Miyashita, Number partitioning on a quantum computer, Physics Letters A 290 (5–6) (2001) 227–233.
[15] F.F. Ferreira, J.F. Fontanari, Probabilistic analysis of the number partitioning problem, Journal of Physics A 31 (1998) 3417–3428.
[16] B. Qu, K. Weng, Path relinking approach for multiple allocation hub maximal covering problem, Computers and Mathematics with Applications 57 (11–12) (2009) 1890–1894.
[17] J. Yang, M. Zhang, B. He, C. Yang, Bi-level programming model and hybrid genetic algorithm for flow interception problem with customer choice, Computers and Mathematics with Applications 57 (11–12) (2009) 1985–1994.
[18] J. Wang, C. Niu, R. Shen, Priority-based target coverage in directional sensor networks using a genetic algorithm, Computers and Mathematics with Applications 57 (11–12) (2009) 1915–1922.
[19] Y. Han, J. Tang, I. Kaku, L. Mu, Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight, Computers and Mathematics with Applications 57 (11–12) (2009) 1748–1755.