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

A R T I C L E   I N F O

A B S T R A C T

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.

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.