METAHEURISTIC ILS WITH PATH RELINKING FOR THE NUMBER PARTITIONING PROBLEM

Cesar Augusto Souza de Oliveira
CEFET-MG, Brazil
E-mail: cesargusto@gmail.com

William de Paula Ferreira
IFSP Campus Suzano, Brazil
E-mail: william.ferreira@ifsp.edu.br

Reinaldo Carlos Mendes
CEFET-MG, Brazil
E-mail: naldo.mendes@gmail.com

Gleisson de Assis
CEFET-MG, Brazil
gleisson.assis@gmail.com

Marcone Jamilson Freitas Souza
CEFET-MG, Brazil
E-mail: marcone.freitas@yahoo.com.br

Submission: 03/01/2017
Accept: 12/01/2017

ABSTRACT

This study brings an implementation of a metaheuristic procedure to solve the Number Partitioning Problem (NPP), which is a classic NP-hard combinatorial optimization problem. The presented problem has applications in different areas, such as: logistics, production and operations management, besides important relationships with other combinatorial problems. This paper aims to perform a comparative analysis between the proposed algorithm with others metaheuristics using a group of instances available on the literature. Implementations of constructive heuristics, local search and metaheuristics ILS with path relinking as mechanism of intensification and diversification were made in order to improve solutions, surpassing the others algorithms.

Keywords: Number Partitioning Problem; NPP; metaheuristics; path relinking; combinatorial Optimization

1. INTRODUCTION

The Partition Problem is: given a set of numbers N, the goal is to divide it in 2 or more subsets (known as partitions) so that the difference between the sums of the numbers inside each partition be the minimum.

The problem's definition sounds very simple but it is a combinatorial optimization problem that belongs to NP-hard class. To a set with *n* numbers, we have 2^n ways to divide these numbers in two or more partitions. In this paper, a bit represents each partition; the bit 0 means the first partition and the bit 1 the second partition. Thus, a feasible solution can be represented by a vector of bits which size is *n*, the position of an element in the solution's vector points to partition associated to the number in the same position of the set that will be allocated.

The Partition Problem can be applied to several areas, such as: logistics, production and operations. Used, for instance, in the scheduling, routing, line balancing and installation projects (H. W. CORLEY, 1971; MACDONALD, 1976). In industry, according to Ducha (2014), a possible application is the raw material partitioning between two machines in the production line, such way that the raw material can be processed faster.

Considering the importance of the Partition Problem, this paper aims to perform a comparative analysis between the proposed algorithm and others metaheuristics to solve a group of instances available on the literature.

The section 2 show a brief literature review, the section 3 contains a detailed description about the problem. Section 4 describe the methodology. In the section 5 and 6 the results will be showed followed by the conclusion and the future works.

2. RELATED WORKS

The works used as motivation to this paper start from studies that reports the difficulty in applying metaheuristics to solve the Partition Problem (ARAGON *et al*., 1991; ARGÜELLO *et al*., 1996; DIAZ et al., 1996; BERRETTA; MOSCATO, 1999). The phenomenon called transition phase is referred by several authors, identified as a specific problem detected in all optimization problem, specially in the Partition Problem (GENT; WALSH, 1995; BERRETTA; MOSCATO, 1999; PERCUS *et al*., 2006; STADLER *et al*., 2003). That consists basically in a transition of a region, where most part of the problem samples has many solutions, to another region, where there is no solution in most cases. Even though this phenomenon has been detected this paper does not address any special focus on it.

3. PARTITION PROBLEM

According to Ducha and Souza (2013), the Partition Problem is one of the first NP-complete problems registered, based on the work of Coork (1971) and Karp (1972). In Garey and Johnson (1979), the problem was classified as one of the six NP-complete classic problems, being it the only one that treats exclusively about numbers.

The structure of the problem and its simplicity makes possible reduce many other problems to the Partition Problem, being referred as nearest neighbor problems. This intensify the need of studies and generalization of existent problems.

4. METODOLOGY

This section presents the detailed information about the implementation. In the section 4.1 is showed how a solution was represented to computational manipulation. In section 4.2 is showed the evaluation function, which is used to do the evaluation of each, generated solution. In section 4.3 is showed the heuristic of refinement, which consists on local search performing movements to explore the neighborhood. In section 4.4 is showed how we generate an initial solution where a random method and a greedy method are proposed. In section 4.5 are shown the ILS metaheuristic, Path Relinking and the proposal algorithm implementing both methods, making our solution a hybridization of them.

To get the computational results we used a set of 25 examples based on samples generated by Berretta and Moscato (1999) with size of 15, 35, 55, 75 and 95 numbers (5 samples for each size) and in all samples, each number has 10 digits. The result analysis is done by arithmetical average of five samples and the main measure of performing is the quality of the solutions in a fixed number and average of the algorithm iterations. These results will be compared with the results from Araujo et al (2004).

The proposed algorithm was implemented in Java and executed on a notebook HP Pavilion, AMD Athlon II Dual-Core P320, 3GB of RAM DDR3, Linux Debian 3.16, Java OpenJDK 7.1, JVM 1.7.0.75.

4.1. Structure of a feasible solution

The data structure to represent a feasible solution is simplest than others problems. The solution consists in, given a set of numbers, determine which partition each number of the set belongs. In the case of two partitions, a bit represents this information; a give number is associated to one or another partition. Considering a set N, and two partitions A and B, we will verify if an element N_1 belongs to A or B and this step will be repeated for others elements of the set N (N_2, N_3… N_n). In this work, the computational representation of a solution is given by a binary tuple of 0 and 1, where a partition A is represented by 0 and the partition B is represented by 1.

Table 1: Data structure

Position	0	1	2	3	4
Numbers	34	67	25	51	13
Solution	1	0	0	1	1

The frame 1 shows in the first line, the position of the two vectors, indicated by the second and third row. Note that the position of the number vector and solution vector are the same. For example, in the second position of numbers we have the element 25, its partition is configured on the second position of the solution vector, which is 0, in the third position we have the element 25 which belongs to partition 1.

4.2. Evaluation function

The Partition Problem can be defined as follow: *Given a set* $A \in \mathbb{N}$*,* or $A \in \mathbb{R}^+$ with elements chosen independently and equally distributed, to partition the set A in k subsets (in this case k = 2) in the way: $D = \{A_1, \dots A_i, \dots A_k\}$, intending to reduce the absolute difference between the sum of the numbers inside each set.

Starting from this formulation we use the model proposed by Karmarkar and Karp (1992) that specifications are comply with this work. Take A_1 as a set with the biggest sum and A_2 the smallest one, then the problem can be formulated by the following equations:

$$\min |z| = \sum_{a_i \in A_1} a_i - \sum_{a_i \in A_2} a_i \qquad (1)$$

$$A_1 \cup A_2 = A \qquad (2)$$

$$A_1 \cap A_2 = \emptyset \qquad (3)$$

$$A \in \mathbb{N} \text{ or } A \in R^+ \tag{4}$$

The restrictions showed in (2), (3) and (4) are commonly omitted, but is important to emphasize that the groups must be mutually exclusives.

4.3. Refinement heuristics

The refinement heuristics normally referred as local search is essentially built on the neighborhood concept, which consists in to generate from a solution s, neighbors s' of this solution by a procedure known as movement. Each solution s' $\in$ N (s), called a neighbor of s, is generated by a modification m in s, called movement. This operation can be represented by s' $\leftarrow s \oplus m.$

An accurate definition of the neighborhood is extremely import to the local search. In fact, starting from a given initial solution must be possible by applying movements reach any neighbor in a finite number of steps.

4.3.1. Movements

We used two movements to generate a neighborhood of *s*. The movements were inversion and swap of bits. The inversion movement consists in each iteration change the bit value of a give position, the value 0 will be changed to 1 and the 1 will be changed to 0, this movement represent the concept of change a number to other partition.

The swap movement consists in change the partition of two given elements, if the partitions are the same nothing is changed, but if the partitions are different, the movement represents the reallocation between partitions. Applying just one movement, we have a new neighbor of the initial solution.

For example, take the solution $s^{(1)}$ = (10011) and $s^{(2)}$ = (10110), starting from the solution $s^{(1)}$ we can by applying the inversion movement to reach $s^{(2)}$ following this path: $^{(1)}$= (10011)$\rightarrow$ (10111)$\rightarrow$(10110) = $s^{(2)}$. In the Partition Problem, the inversion movement is not enough to explore all the neighborhood, we need to apply both movements (inversion and swap) to guarantee all neighbor of *s* can be generated.

4.3.2. Local search

The local search procedure implemented was the Best Improvement Method, which consists in evaluate by an evaluation function all the solutions *s'* generated from an initial solution by applying the movements on it and return the best solution. If a solution better than the initial solution is found, it replaces the initial solution, when no better solution is found we can say that a local optimum related to the neighborhood was found.

4.4. Generation of initial solution

Some metaheuristics need more than others start from a good initial solution to converge to a good solution as well.

We choose two distinct ways to generate an initial solution: the first is a random initial solution, which means to distribute the elements between the partitions randomly. We consider the fact that if a solution has just one partition it is not accepted as a feasible solution.

The second way to generate an initial solution is more careful and starts from a greedy constructive idea specific to the Partition Problem; the local search is applied to the generated solution to obtain the initial solution. The greedy method consists in sort the numbers downward and distribute them (following the descending order) between the partitions, the selected partition is that have the smaller sum of numbers allocated on it.

This method works like placing stones, gravel and then sand in a container, this order of arrangement of the elements causes the gravel to permeate the vacant spaces between the stones and finally the finer sand as the gravel permeates the remaining small spaces occupied by the gravel, so all the space is utilized minimizing the final space required to contain all these materials.

For example, consider the set: (36, 67, 25, 51, 13), first sort the set downward, we have: (67,51,34,25,13). The next step is start the distribution considering the two partitions A and B, the partition A receives the first number, then A = {67}, the second number goes to the smaller partition, in this case B because the sum is equals to zero (the partition is empty) , B = {51}, the third element (34) goes to B which still being the smaller partition (the sum of elements, just one, is 51) then B = {51, 34} at this moment the partition B has the sum equals to 85 and the partition A has the sum equals to 67, the next element (25) is allocated in partition A which turns in the bigger partition with the sum equals to 92. Finally B receives the last element 13.

The greedy method generates the solution: A = {67, 25} and B = {51, 34, 13}. Evaluating this solution, we have z = | 92 – 98 |, z = 6. After generate the initial solution the local search is applied on it aiming find a local optimum, in our example the solution will be the same of the greedy constructive method.

4.5. The proposal algorithm: ILS-PR

The proposal algorithm ILS-PR is a hybridization of the Iterated Local Search (ILS) and Path-Relinking. The hybridization idea consists in improve the results generated by just ILS metaheuristic.

4.5.1. ILS

The ILS method (Fig. 1) is based on a local search with a procedure to generate new solutions by perturbations on the local optimum, opening chances to find other better local optimum by applying movements exploring the search space. The ILS must contain (i) a procedure to generation an initial solution, (ii) a local search procedure to improve the initial solution, (iii) a procedure to disturb the current solution *s* leading it to an intermediate solution *s'*.

The perturbation must be strong enough to find out a way to escape from the local optimum traps, and finally, (iv) an acceptance criteria which decides the type of disturbing will be applied to solution in the next algorithm iteration.

```
ILS procedure
1   $s_0$ ←
2   $s$ ← LocalSearch($s_0$)
3   WHILE (stop criteria are not match) DO
4       $s'$ ← disturbance(historic,s);
5       $s''$ ←LocalSearch(s');
6       $s$ ←AcceptanceCriteria(s,s'',historic);
7   end while;
    end ILS
```

Figure 1: Iterated Local Search (ILS) Algorithm

We used as acceptance criteria: move to the local optimum only if it is smaller than the current local optimum, which means, only if f(s'') < f(s), in this case a minimization problem.

The Figure 2 shows what occurs when the perturbation is made on an initial solution *s*, a second solution is returned and the local search procedure on s' returns a new local optimum that in our example is the global optimum.

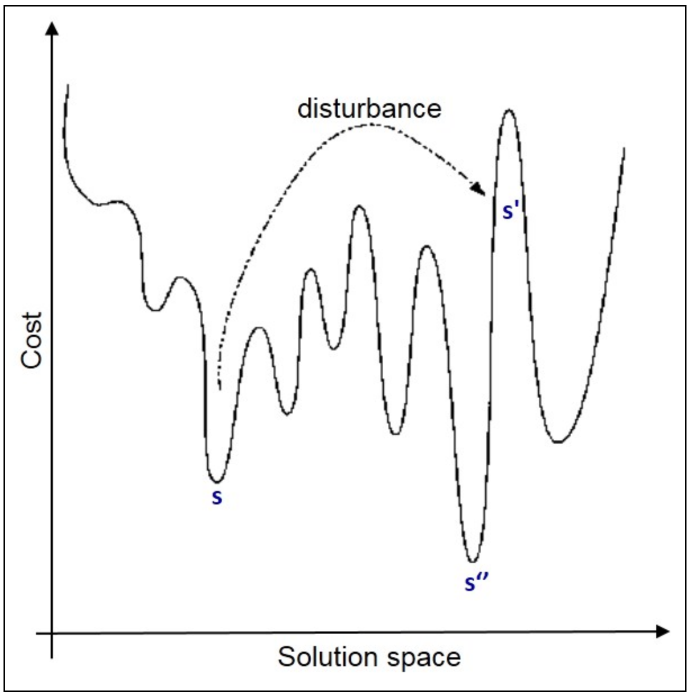

Figure 2: Schematic representation of ILS operations

We used three levels of perturbations: the level one consists in changing the bit in two different positions together in current solution, these two positions are defined randomly, for example: consider a solution s = (10011) the numbers 1 and 3 selected randomly, the result of perturbation is s' = (11001). The second level consists in a bit inversion with a swap between two bits, all bits are selected randomly. Finally, the third level consists in swapping two bits, the two positions are generated randomly, the reference bits are swapped with subsequent or previous bits if one or both are the last.

In case of the selected bit is equals to its subsequent an inversion movement in both will be performed. Each level brings an abrupt perturbation characteristic, it is necessary to the algorithm has enough strength to be able to escape from a deeper valley with larger jumps and subtlety to smaller jumps to reach closer valleys.

4.5.2. Path Relinking

The Path Relinking technic was proposed as an intensification strategy to explore trajectories, which connect elite solutions obtained by the Tabu Search method. This search for better solutions consists in generate and explorer paths in the solutions space starting from one or more elite solutions and leading to another elite solution. To do this, movements are selected which insert attributes from lead solution into current solution. In this way, path relinking can be seen as a strategy that aims incorporate attributes of good solutions, favoring the selection of movements that contain them.

We applied path relinking as an intensification of each local optimum generated after the local search phase. This strategy is known by get better results from this method. The method can be used as a post-optimization strategy looking for better solutions connecting paths between solutions from an elite group already ranked.

In our algorithm path relinking consists in create an elite group containing five solutions with a good quality which are going ranked during the iterations of any previous other algorithm of generating solutions. The solutions contained in elite group must have at least 20% of diversity between them; a new solution is inserted in the elite group if it is better than the worse solution inside the group.

The path relinking procedure itself consists of transforming an elite solution, selected randomly or sequentially, into the current solution, copying from the current solution in each iteration, an element from the elite solution, after each replacement of an element of the elite solution by an element in the same position of the current solution the local search procedure is performed to determine a local optimum starting from the path proposed by the method (Figure 3).

It is very important to emphasize that the local search procedure, unlike the conventional procedure, should not apply the movement(s) in the position where the substitution occurred. This procedure runs from the current solution to all solutions of the elite group looking for an improvement solution.

```
Path relinking procedure
$\bar{g} \leftarrow s$
Assign to g' the best solution between s and g;
Calc the set of possible movements $\Delta(s,g)$;
WHILE $|\Delta(s,g)| \neq 0$ DO
    Assign to g'' the obtained best solution using the best movement $\Delta(s,g)$ to $\bar{g}$;
    Exclude from $\Delta(s,g)$ this movement
    $\bar{g} \leftarrow g''$;
    IF f($\bar{g}$) < f(g') THEN
        $g' \leftarrow \bar{g}$;
    end IF
end WHILE
Return g';
```

Figure 3: Path Relinking procedure

4.5.3. ILS-PR

In this work, Path Relinking (PR) was used as an intensification and diversification strategy in each iteration of ILS, as shown in the Fig. 4.

The first line points to the generation of an initial solution s_0 which in our algorithm is done by a greedy constructive heuristic shown in (4.4). The second line is the local search phase generating. While a stop condition is not reached s' receives a perturbation performed on the solution s with a pre-defined perturbation level. The solution s'' receives the resulting solution from a local search applied in s' pointing to a local optimum which can be added to elite group or not and update or not the solution s*.

In the line 7 PR is called who will iterate from s'' with all elite solutions, if there is an improvement the solution s is updated and perturbation level reduced or kept (in case of minimum level), on the other hand the perturbation level is increased and the processed start again from the line 4.

```
1     $s_0$ ← build_Inicial_Solution ( )
2     $s$ ← BL ($s_0$)
3     WHILE (cpna) DO
4         $s'$ ← disturbance(s, level)
5         $s''$ ← BL(s')
6         Update CE
7         Update s*
8         PR (s'', CE)
9         IF  f(s'') < f(s) THEN
10            $s$ ← $s''$
11            Level = 0
12        ELSE
13            Level ++
14    end
```
Figure 4: ILS with Path Relinking procedure

5. RESULTS AND ANALISYS

The results presented here aims to compare just the quality of the solutions. Frame 1 show the methods applied by Araújo (2004) while the last two columns show our results. The first column defines the size of the instances and the other abbreviations has the meaning as follow: Hill-Clibing (HIC), Randon Generated Test (RGT), Simulated Annealing (SA), Tabu Search with Short Term Memory (TSSTM), Genetic Algorithms (GA), Memetic Algorithms (MA), Iterated Local Search (ILS), Iterated Local Search with Path Relinking (ILS+PR).

Frame 1: results comparison

Inst.	HIC	RGT	SA	TSSTM	GA	MA	ILS	ILS+PR
15	89.453.562,0	1.048.865,2	1.048.865,2	1.048.865,2	1.048.865,2	1.048.865,2	998.680,8	**509.665,0**
35	32.722.712,0	252,2	114,6	340,2	3.692,0	**53,8**	49.071,2	2.828,0
55	7.834.749,0	218,6	234,6	**101,00**	3.963,4	227,4	10.22,6	505,0
75	3.328,397,0	312,4	347,2	336,0	4.816,8	212,4	3.196,0	**198,0**
95	1.585.882,00	386,8	135,2	610,8	4.886,4	108,8	1.131,0	**60,0**

Table 1 shows the gray-bottomed cells the best results found by the algorithms. The results of ILS+PR surpassed all other algorithms for the 15, 75 and 95 numbers and were surpassed by the Memetic Algorithm (MA) in the 35-number instance and by the Short-Term Tabu Search Algorithm (STTS) in Instance of 55 numbers. Our algorithm presented worsening in the transition phase region where it is already expected to be more difficult to obtain good results.

6. CONCLUSIONS AND FUTURE WORKS

This work proposed an algorithm to solve the Partition Problem using the metaheuristics Iterated Local Search (ILS) with Path Relinking (PR) as strategy of diversification and intensification. The computational results obtained by the proposed algorithm were compared with the results of (ARAUJO, 2004).

Analyzing these results, we showed that the algorithm ILS+PR surpassed the others algorithms in 3 of 5 tested instances. In future works we suggest improve the constructive heuristic strategy, since the used method was not sufficient to generate goo initial solutions due to the small diversity of the numbers contained in the samples, all with ten digits, that is, very large and relatively close numbers.

A more specific approach to the difficulty encountered in the transition phase will also be welcome. Another important contribution will be to allow a quantity *k* of partitions as well as to present run-time studies of the algorithm for more comprehensive comparisons.

7. ACKNOWLEDGMENTS

The authors thank the Graduate Program in Mathematical Modelling and Computational (PPGMMC) at CEFET-MG for the support in the development of this work.

REFERENCES

ARAGON, C. R.; JOHNSON, D.; MCGEOCH, L.; SCHEVON, C. (1991) Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. **Operations Research**, v. 39, n. 3, p. 378–406.

ARGÜELLO, M. F.; FEO, T. A.; GOLDSCHMIDT, O. (1996) Randomized methods for the number partitioning problem. **Computers & Operations Research**, v. 23, n. 2, p. 103–111.

BERRETTA, R.; MOSCATO, P. (1999) **The number partitioning problem**: an open challenge for evolutionary computation? In New ideas in optimization. McGraw-Hill Ltd., UK, p. 261-278.

CITESEER. KARP, R. M. (1972) **Reducibility among combinatorial problems**. Springer.

COOK, S. A. (1971) The complexity of theorem-proving procedures. **In Proceedings of the third annual ACM symposium on Theory of computing**, pages 15 1–158. ACM.

ARAUJO, S. A.; CONSTANTINO, A. A.; MENDONÇA NETO, C. F. X. (2004) Meta-heurísticaspara o problema de partição de números. **XXXVI – SBPO**. São João del-Rei.

DIAZ, A.; GLOVER, F.; GHAZIRI, H.; GONZÁLEZ, J.; LAGUNA, M.; MOSCATO, P.; AND TSENG, F. (1996) Optimización heurıstica y redes neuronales. **Editorial Paraninfo**, n. 235, p. 158–159.

DUCHA, F. A.; DE SOUZA, S. R. (2013) Algorithms analysis for the number partition problem. **XXXIV CILAMCE**.

DUCHA, F. A. (2014) **Algoritmos e modelos para solução do problema de partição de números**. Dissertation (Master in Mathematical and Computational Modeling). Belo Horizonte: MMC/CEFET-MG, Available at: http://www.files.scire.net.br/atrio/cefet-mg-ppgmmc_upl/THESIS/193/fernando_andrade_ducha.pdf. Access: 22/12/2016.

GAREY, M. R.; JOHNSON, D. S. (1979) **Computers and intractability**: a guide to the theory of np-completeness. San Francisco, LA: Freeman.

GENT, I. P.; WALSH, T. (1995) **The number partition phase transition**. Department of Computer Science, University of Strathclyde.

H. W. CORLEY; ROBERTA JR., S. D. (1972) A Partitioning Problem with Applications in Regional Design. **Operations Research**, v. 20, n. 5 (Sep. - Oct.), p. 1010-1019.

HOFFMAN, K.; PADBERG, M. (2008) Set covering, packing and partitioning problems Set Covering, Packing and Partitioning Problems. **Encyclopedia of Optimization**, Springer US, p. 3482-3486.

LARSEN J. **Set Partitioning and Applications**. Available at: http://www2.imm.dtu.dk/courses/02735/sppintro.pdf. Access: 09/10/2016.

MACDONALD, P. (1976) Combinatorial Programming: Methods and Applications. **Journal of the Operational Research Society**, v. 27, n. 3, p. 640-640.

PEDROSO, J. P.; KUBO, M. (2010) Heuristics and exact methods for number partitioning. **European Journal of Operational Research**, v. 202, n. 1, p. 73-81.

PERCUS, A.; ISTRATE, G.; MOORE, C. (2006) **Computational complexity and statistical physics**. Oxford University Press..

STADLER, P. F.; HORDIJK, W.; FONTANARI, J. F. (2003) Phase transition and landscape statistics of the number partitioning problem. **Physical Review**, v. 67, n. 5, p. 056701.