Mrs. P. Vasanthi, *et al*, International Journal of Advances in Agricultural Science & Technology,
Vol.8 Issue 8, Aug 2021, pg.210-213

ISSN:2348-1358
Impact Factor: 6.901
NAAS Rating: 3.77

Performance Analysis of Sequential and Parallel GRASP Algorithms for the 0-1 Multidimensional Knapsack Problem

Mrs. P. Vasanthi
Assistant Professor, Department of CSE,
Malla Reddy College of Engineering for Women.,
Maisammaguda, Medchal., TS, India

Abstract:

The knapsack problem is a widely known problem in combinatorial optimization and has been object of many researches in the last decades. The problem has a great number of variants and obtaining an exact solution to any of these is not easily accomplished, which motivates the search for alternative techniques to solve the problem. Among these alternatives, metaheuristics seem to be suitable on the search for approximate solutions for the problem. In this work we propose a sequential and a parallel implementation for the multidimensional knapsack problem using GRASP metaheuristic. The obtained results show that GRASP can lead to good quality results, even optimal in some instances, and that CUDA may be used to expand the neighbourhood search and as a result may lead to improved quality results.

Keywords:

GRASP, 0-1 multidimensional knapsack problem, CUDA.

Introduction

The knapsack problem (KP) is one of the most known problems of combinatorial optimization. Informally, given a set of items and a knapsack with a known maximum capacity, we want to fill the knapsack with the most valuable subset of the items subject to the knapsack capacity. The 0-1 multidimensional knapsack problem (MKP) is a generalization of KP which can have more than one constraint and the items cannot be partitioned. In this variation, the traditional approaches to solve KP cannot be efficiently applied anymore and this motivates the search for alternative approaches. Formally, we can define MKP as, given a set of n items with their respective values and m > 1 constraints, deciding which items should be placed inside the knapsack aiming to maximize its value without extrapolate any of the constraints. We can formulate the problem using the

Equation 1 [2]:

$$\max\{\sum_{j=1}^{n} v_j x_j\}, j = 1, ..., n$$

$$\text{subject to:} \sum_{j=1}^{n} r_{ij} x_j \leq b_i, i = 1, ..., m$$

where V = {v1, v2, ..., Vn} is an array containing the item values, B = {b1, b2, ..., bm} is an array with the constraint's values and matrix R = {r11, ..., r1m, ..., rm1, ..., rmn} stores how much each constraint is used by each item. A solution to the problem is a configuration of array X = {x1, x2, ..., xn}, which satisfies every constraint of array B, where each xi can assume the values 1 or 0 indicating, respectively, the corresponding item belongs or not to the resulting knapsack. This solution cannot be easily obtained, in fact, this is an N P-hard problem. Different metaheuristics have been applied to the solution of the 0-1 MKP [10], some works deserve to be highlighted and their results are compared to the ones obtained in our implementstations: the genetic algorithm proposed by Chu and Beasley [2], the parallel implementation based on ant-colony optimization, proposed by Finger et. al [8] and the neurogenetically algorhythm proposed by Deane and Agarwal [5]. In previous works [3, 4] we had already presented sequential and parallel implementations of genetic algorithms and augmented neural networks to find approximated solutions to 0-1 MKP and in this work we propose

ISSN:2348-1358
Impact Factor: 6.901
NAAS Rating: 3.77

sequential and parallel GRASP approaches to accomplish this task. The rest of this text is organized as follows. Section 2 presents the definition of GRASP metaheuristic. Section 3 describes our proposed solution. In Section 4 we present the obtained results compared to other approaches. Section 5 shows our conclusions and ideas for future works.

Greedy Randomized Adaptive Search Procedure

Greedy Randomized Adaptive Search Procedure (GRASP) [9] is a multi-start metaheuristic that performs a sequence of iterations, each one consisting of two phases: construction and local search. The construction phase is responsible for building an initial feasible solution using a greedy strategy and a restricted candidate list (RCL); the local search explores the neighbourhood of this solution until a local minimum (or maximum) is found. The result of the whole process is the best overall solution. GRASP has been successfully applied to solve several optimization problems and for further reading on GRASP heuristic we suggest the annotated GRASP bibliography assembled by Festa and Rezende [7].

The Proposed Solution

In this section we present our approach to use GRASP to solve the 0-1 MKP. We implemented a sequential version and a version using GPGPU with CUDA library. The sequential impelmentation was designed following the basic steps proposed by Rezende and Ribeiro [9]; the RCL is constructed using the quality-based policy. In order to evaluate the quality of the items, we use the pseudo utility of each item which is computed based on its value and on its demand for each one of the constraints, as stated in [4, 6]. The RCL will contain the items with higher pseudo utilities values. The pseudocodes of our GRASP main procedure and of Construct Solution() are shown in Algorithms 1 and 2, where the parameter α is used to guide the assembly of the RCL.

Algorithm 1: GRASP_MKP(maxIterations, α)

```
1  Read_Input;
2  BestSolution ← ∅;
3  for k ← 1, maxIterations do
4  |  Solution ← ∅;
5  |  Solution ← ConstructSolution(Solution, α);
6  |  Solution ← LocalSearch(Solution, α);
7  |  if Solution is better than BestSolution then
8  |  |  BestSolution ← Solution;
9  |  end
10 end
11 return BestSolution;
```

Algorithm 2: ConstructSolution(Solution, α)

```
1 repeat
2 |  Calculate the pseudoutilities of the items;
3 |  Construct RCL and randomly choose one item e from RCL;
4 |  if e can be added to Solution then
5 |  |  Solution ← Solution ∪{e};
6 |  end
7 until e can not be added to Solution;
8 return Solution;
```

Local search, implemented using the ideas presented in [11], works iteratively aiming to find a better solution, it repeats the process of removing some items from current solution until any available item can be incorporated to the solution and then a new complete solution is reconstructed. An auxiliary n-position Boolean array (named marked) is used to guide this process, it stores information that helps to keep track of which was the first item removed in each iteration. The steps of local search, based on [11], are listed in Algorithm 3. Our CUDA implementation aims to expand the neighbourhood search using different threads to iteratively construct different initial solutions in parallel and then execute local search in their own generated solution, also in parallel. This approach can be seen as many parallel executions of the sequential program; the CPU is used to manage the iterations and to find the best solution achieved at the end of the whole process.

Implementation and Comparative Results

ISSN:2348-1358
Impact Factor: 6.901
NAAS Rating: 3.77

The implemented programs were executed using the instances of ORLIB library [1], one of the most used set of instances in works related to the problem. The ORLIB library is composed of a set of 270 test instances considering 5, 10 or 30 constraints and 100, 250 or 500 items. The gap is a key concept to evaluate the quality of the obtained results, it is defined as the percentage of the difference between the values of the obtained solution and the best-known solution. Both our programs were executed 30 times for each test instance and their medium gaps and times were computed, as well as the standard deviations of the obtained gaps. Each execution of the sequential program consisted of 1000 iterations, while the CUDA version executed 100 iterations. The number of threads blocks were set to 100, except for the instances with 500

Algorithm 3: LocalSearch(Solution, α)

```
1  Initialize array marked;
2  Copy Solution to CurSol;
3  while there is any not marked item do
4      Calculate the pseudoutilities of the items;
5      repeat
6          Find the e ∈ CurSol with the smallest pseudoutility and remove it from CurSol;
7      until all the available items may be added to the solution;
8      ConstructSolution(CurSol, α);
9      if CurSol is better than Solution then
10         Solution ← CurSol;
11         Update array marked;
12     end
13     else
14         CurSol ← Solution;
15         First item removed is marked;
16     end
17 end
18 return Solution;
```

items which used 20 threads blocks. The obtained results with every configuration were less than 1% inferior to the best-known solution values. Only 14 test instances had more than 1% of medium gap, but these values did not exceed 1.2%. In order to demonstrate how promising GRASP metaheuristic is to solve the 0-1 MKP, we compared the obtained results with the ones from Chu and Beasley's genetic algorithm [2], Deane and Agarwal's neurogenetically approach [5] and Finger's et al. ant-colony algorithm [8]. Table 1 shows the comparisons of the medium gaps of the test instances grouped by number of items and Table 2 shows the medium gaps grouped by number of constraints.

Table 1: Comparison of the average gap with the test instances grouped by number of items.

n	GA	Neurogenetical	ACO	SeqGRASP	CudaGRASP
100	1.0744	1.0800	1.0889	0.3017	0.2414
250	0.3744	0.3780	0.8522	0.2343	0.2607
500	0.1800	0.1860	0.5389	0.2074	0.2456

Table 2: Comparison of the average gap with the test instances grouped by number of constrains.

m	GA	ACO	SeqGRASP	CudaGRASP
5	0.2611	0.2133	0.0817	0.0514
10	0.4622	0.6511	0.1913	0.1593
30	0.9056	1.6156	0.4703	0.5369

By the values shown in Table 1 we can see that GRASP leads to smaller gaps than the other approaches, except for the instances with 500 items where the resulting gaps are slightly greater than the obtained by Chu and Beasley's genetic algorithm and by Deane and Agarwal's neurogenetically approach. Analysing the values in Table 2 we see that GRASP outperforms the other approaches, although we could not compare with the results from the neurogenetically algorithm because these values were not available on the reference work.

Conclusion

In this work we presented sequential and GPGPU-based algorithms using GRASP metaheuristic to solve 0-1 MKP and tested them using the test instances of ORLIB library. The tests results showed GRASP is a promising alternative, achieving solutions less than 1% inferior to the best-known solutions. When compared to other metaheuristics implementations, such as the ones in

ISSN:2348-1358
Impact Factor: 6.901
NAAS Rating: 3.77

[2], [5] and [8], GRASP also obtained better quality results in almost every test configuration. The GPGPU algorithm achieved slightly better results than the sequential version, mainly because it allowed a more effective neighbourhood exploration. Nevertheless, this improvement in quality also led to an increase in execution time. As future works, we will study and implement other sequential and parallel metaheuristics to solve 0-1 MKP and compare them to GRASP, specially simulated annealing and variable neighbourhood search (VNS). We will also evaluate how GRASP can be applied to other Optimutation problems, such as the traveling salesman problem (TSP) and the quadratic assignment problem (QAP).

References

[1] J. E. Beasley. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society, 41(11):1069–1072, 1990.

[2] P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics, 4(1):63–86, June 1998.

[3] B. A. Dantas and E. N. C´aceres. Implementa¸c˜oes paralelas para o problema da mochila multidimensional usando algoritmos gen´eticos. In Anais do XLVI Simp´osio Brasileiro de Pesquisa Operacional, 2014.

[4] B. A. Dantas and E. N. C´aceres. A parallel implementation to the multidimensional knapsack problem using augmented neural networks. In Proceedings of the 2014 Latin American Computing Conference (CLEI), pages 570–578, 2014.

[5] J. Deane and A. Agarwal. Neural, genetic, and neurogenetic approaches for solving the 0-1 multidimensional knapsack problem. International Journal of Management & Information Systems - First Quarter 2013, 17(1):43–54, 2013.

[6] J. Deane and Anurag Agarwal. Neural metaheuristics for the multidimensional knapsack problem. Technical report, 2012.

[7] P. Festa and M. G. C. Resende. An annotated bibliography of GRASP – part ii: Applications. International Transactions in Operational Research, 16(2):131–172, 2009.

[8] H. Fingler, E. N. C´aceres, H. Mongelli, and S. W. Song. A CUDA based solution to the multidimensional knapsack problem using the ant colony optimization. Procedia Computer Science, 29(0):84 – 94, 2014. 2014 International Conference on Computational Science.

[9] C. C. Ribeiro M. G. C. Resende. Greedy randomized adaptive search procedures. In Fred Glover and Gary A. Kochenberger, editors, Handbook of Metaheuristics, pages 219–249. Kluwer, 2002.

[10] M. Varnamkhasti. Overview of the algorithms for solving the multidimensional knapsack problems. Advanced Studies in Biology, 4(1):37–47, 2012.

[11] D. S. Vianna and M. F. D. Vianna. Local search-based heuristics for the multiobjective multidimensional knapsack problem. In Produ¸c˜ao, volume 23, pages 478–487. 2013.