Special Issue of International Journal of Computer Applications (0975 – 8887)
on Optimization and On-chip Communication, No.1. Feb.2012, www.ijcaonline.org

A New Bit Wise Technique for 3-Partitioning Algorithm

Rajkumar Jain
Indian Institute of Technology, Indore,
Survey No.113/2-B, Opp. to Veterinary College
AB- Road, Village Harnia Khedi, Mhow (M.P.)

Narendra S. Chaudhari
Indian Institute of Technology, Indore,
Survey No.113/2-B,Opp. to Veterinary College
AB- Road, Village Harnia Khedi, Mhow (M.P.)

ABSTRACT

From last five decades peoples are working on Partitioning problem. Partitioning is a fundamental problem with applications in several fields of study. The 3-partition problem is one of the most famous strongly NP-complete combinatorial problems in computer science [1]. Most of the existing partitioning algorithms are heuristics in nature

General Terms

Partitioning, 3-Partitioning, NP-Complete Problem, Stirling number, combinatorial optimization, Pattern Matching.

Keywords

3-BitPartition, Rule of Belongingness, Optimal Partitioning technique.

1. INTRODUCTION

A Partition of a set *U* is a subdivision of the set into subsets that are disjoint and exhaustive, i.e. every element of *U* must belong to one and only one of the subsets. The subsets A_i in the partition are called cells. Thus $\{A_1, A_2, . . ., A_r\}$ is a partition of *U* if two conditions are satisfied: (1) $A_1 \cap Aj = \phi$ if $i \neq j$ (the cells are disjoint) and (2) $A_1 \cup A_2 \cup \ldots \cup A_r = U$ (the cells are exhaustive). In computer science, the partition problem is an NP-complete problem [2] and it is also NP-Hard to find good approximate solutions for this problem [3].

1.1 Combinatorial problems

Combinatorics is, loosely, the science of counting. This is the area of mathematics which deals with the study of families of sets (usually finite) with certain characteristic arrangements of their elements or subsets, and ask what combinations are possible, and how many there are.

Combinatorics is a branch of mathematics concerning the study of finite or countable discrete structures. Combinatorial Analysis is a branch of mathematics which teaches to exhibit all the possible ways in which a given number of things may be associated and mixed together; it is to ensure to enumerate all collection or arrangement of the things. Combinatorics is concerned with arrangements of the objects of a set into patterns satisfying specified rules.

Combinatorics is a branch of mathematics concerning the study of finite or countable discrete structures. Aspects of combinatorics include counting the structures of a given kind and size, deciding when certain criteria can be met, and constructing and analyzing objects meeting the criteria, finding "largest", "smallest", or "optimal" objects. Combinatorics has many applications in optimization, computer science and in statistical physics.

1.2 Combinatorial optimization problems

In applied mathematics and theoretical computer science, combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects.

In many such problems, exhaustive search is not feasible. It operates on the domain of those optimization problems, in which the set of feasible solutions is discrete or can be reduced to discrete, and in which the goal is to find the best solution. Some common problems involving combinatorial optimization are the traveling salesman problem, minimum spanning tree problem and Cluster Analysis

Combinatorial optimization is a subset of mathematical optimization that is related to operations research, algorithm theory, and computational complexity theory. It has important applications in several fields, including artificial intelligence, machine learning, mathematics, and software engineering.

The optimization version asks for the "best" partition, and can be stated as: Find a partition into two subsets S1, S2 such that Max(Sum(S_1), Sum(S_2)) is minimized.

1.3 Application of Partitioning

Application of Partitioning problem [4, 5] is very broad. Following are the important applications of partitioning problem:

1. In circuit and in VLSI design where graph or network decompositions are of great interest.
2. In parallel processing
3. In combinatorial optimization such as simulated annealing, mean field annealing, Tabu Search, genetic algorithms, and stochastic evolution
4. In many DB application, including histogram-based selectivity estimation, load balancing and construction of index structures.
5. The scheduling of jobs to processors so as to minimize some cost function is an important problem in many areas of computer science.
6. Optimal generalized Block Distribution in parallel Computing
7. In cluster Analysis

2. 3-PARTITION PROBLEM

The 3-Partition problem [8] is one of the most famous strongly NP-complete combinatorial problems in computer science.

In this variation of 3-partition problem, set S must be partitioned into |*S*|/3 triples each with the same sum. The problem is to decide whether a given multi-set of integers can be partitioned into triples that all have the same sum.

In the 3-Partition Problem, we are given a positive integer *b* and a set *N* = { 1, 2, 3, ……, *n* } of *n* = 3*m* elements, each having a positive integer size a_j , such that .

$$\sum_{j=1}^{n} a_j = m_b$$

The problem is to determine whether a partition of *N* into *m* subsets, each containing exactly three elements from N and such that the sum of the sizes in each subset is *b*, exists.

The solution is yes if such a partition exists, and no otherwise. Garey and Johnson (1975) [1] originally proved that 3-partition to be NP-complete.

2.1 Complexity of Partitioning Problem

Zuckerman shows that all of the problems (21 Problems) listed by Karp's [9] are NP-Complete and have a version that's hard to approximate. Partitioning Problem is one of the problems in the listing of Karp's 21 Problems.

These versions are obtained from the original problems by adding essentially the same, simple constraint. These problems are absurdly hard to approximate. Karp also shows that one cannot even approximate $\log^{(k)}$ of the magnitude of these problems to within a constant factor, where $\log^{(k)}$ denotes the iterated logarithm, unless NP is recognized by slightly super polynomial randomized machines.

2.2 Stirling numbers

In mathematics, Stirling numbers [7,8] arise in a variety of combinatorics problems.

They are named after James Stirling, who introduced them in the 18th century. Two different sets of numbers bear this name: the Stirling numbers of the first kind and the Stirling numbers of the second kind.

2.2.1 Stirling numbers of the first kind

They commonly occur in combinatorics, where they appear in the study of permutations. Stirling numbers of the first kind [7, 8] are written with a small *s.*

The Stirling Numbers of the First Kind can be defined as s(n, k) ways of partitioning a set of n elements into *k* disjoint cycles. Stirling numbers of the first kind counts the number of ways to arrange *n* objects into *k* cycles instead of subsets. Stirling numbers of the first kind are represented as

$\begin{bmatrix} n \\ k \end{bmatrix}$ it Means "*n* cycle's *k*".

Cycles are cyclic arrangements, therefore
$\{a, b, c, d\} = \{b, c, d, a\} = \{c, d, a, b\} = \{d, a, b, c\}$
But $\{a, b, c, d\} \neq \{a, b, d, c\}$

There are eleven different ways to make two cycles from four elements (see table 1).

Table 1: Number of Combination generated for Stirling numbers of the first kind s(4,2)

No. of Combination	Partition 1
1	{*a*, *b*, *c*}, {*d*}
2	{*a*, *b*, *d*}, {*c*}
3	{*a*, *c*, *d*}, {*b*}
4	{*b*, *c d*}, {*a*}
5	{*a*, *c*, *b*}, {*d*}
6	{*a*, *d*, *b*}, {*c*}
7	{*a*, *d*, *c*}, {*b*}
8	{*a*, *b*, *c*}, {*d*}
9	{*a*, *b*}, { *c*, *d*}
10	{*a*, *c*}, { *b*, *d*}
11	{*a*, *d*}, { *b*, *c*}

Recursive relation for Stirling numbers of the first kind is

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}$$

Where $\begin{bmatrix} n \\ k \end{bmatrix} = | \, s(n,k) \, |$

For *k* > 0, with the initial conditions

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1 \text{ and } \begin{bmatrix} n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} = 0 \text{ for } n > 0$$

2.2.2 Stirling number of the second kind

Stirling numbers of the second kind [7, 8] occur in the field of mathematics called Combinatorics and in the study of partitions. In mathematics, particularly in combinatorics, a Stirling number of the second kind is the number of ways to partition a set of n objects into k non-empty subsets and is denoted by S(n,k). Stirling numbers of the second kind are written with a Capital *S.*

Recursive relation for Stirling numbers of the second kind is

$$\begin{Bmatrix} n+1 \\ k \end{Bmatrix} = k \begin{Bmatrix} n \\ k \end{Bmatrix} + \begin{Bmatrix} n \\ k-1 \end{Bmatrix}$$

Where $\begin{Bmatrix} n \\ k \end{Bmatrix} = | \, s(n,k) \, |$

for *k* > 0, with the initial conditions

$$\begin{Bmatrix} n \\ k \end{Bmatrix} = 1 \text{ When } n = k \text{ and } \begin{Bmatrix} n \\ 1 \end{Bmatrix} = 1 \text{ and for } n > 0$$

Example: In how many ways we can partition 4 objects into 2 classes.
Let us Consider a set *U*, such that *U* = { *a*, *b*, *c d* } | *U* | = 4.

The number of ways in which we can partition 4 objects into 2 classes is denoted by $\begin{Bmatrix} 4 \\ 2 \end{Bmatrix}$ and is called a Stirling number of the second kind.

$\begin{Bmatrix} 4 \\ 2 \end{Bmatrix}$ = 7, see Table 2 for details

Table 2: Number of Combination generated for Stirling numbers of the second kind S (4, 2)

No. of Partition	Partition 1	Partition 2
1	a, b	c, d
2	a, c	b, d
3	a, d	b, c
4	a	b, c, d
5	b	a, c ,d
6	c	a, b, d
7	d	a, b, c

2.2.3 *Bell Number[4,5]*

The Bell numbers describe the number of ways a set with *n* elements can be partitioned into disjoint, non-empty subsets. Starting with $B_0 = B_1 = 1$ bell numbers are:

1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597,

In general, B_n is the number of partitions of a set of size n. A partition of a set S is defined as a set of nonempty, pair-wise disjoint subsets of S whose union is S. For example, $B_3 = 5$ because the 3-element set {*a*, *b*, *c*} can be partitioned in 5 distinct ways: The set {*a*, *b*, *c*} can be partitioned in the following ways(see Table 3).

Table 3: Number of partition generated for B_3 Other way of Representation of B_3

No. of Partition	Partitions
1	{ {*a*}, {*b*}, {*c*} }
2	{ {*a*, *b*}, {*c*} }
3	{ {*a*, *c*}, {*b*} }
4	{ {*b*, *c*}, {*a*} }
5	{ {*a*, *b*, *c*} }

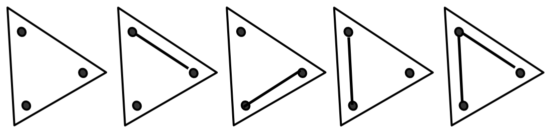

Fig 1: Number of partition generated for B_3

So, $B_3 = 5$

Bell number satisfies the recurrence formula

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

Each Bell number is a sum of Stirling numbers of the second kind

$$B_n = \sum_{k=1}^{n} \begin{Bmatrix} n \\ k \end{Bmatrix}$$ Where $\begin{Bmatrix} n \\ k \end{Bmatrix}$ is a Stirling numbers of the second kind $\begin{Bmatrix} n \\ k \end{Bmatrix} = |S(n, k)|$

For Example: $B_3 = \sum_{k=1}^{3} \begin{Bmatrix} 3 \\ k \end{Bmatrix}$

$B_3 = S(3, 1) + S(3, 2) + S(3, 3)$
$B_3 = 1 + 3 + 1$
$B_3 = 5$

3. PROPOSED APPROACH FOR 3-PARTITIONING - BIT WISE REPRESENTATION OF PARTITION

In this method Partitions are represented in the form of bits. For the 3 partitioning of n-entities 3n bits are required. Each of the n bits represents one partition. When 3-partitioning is represented in bit form, a specific pattern is generated for 3-Partition for any value of n. In this pattern first *n* consecutive bits represent 1st Partition, next n consecutive bits represent 2nd Partition and last n consecutive bits represent 3rd Partition.

3.1 Rule of Belongingness

If an entity O_i belongs to Partition C_i then the *i*th value of O_i is 1 otherwise it is zero.

Let us consider a set of 5 entities *U* = {*a*, *b*, *c*, *d*, *e*},

Let us a take a sample

$P_1 = \{a, b, c\}$, $P_2 = \{e\}$ & $P_3 = \{d\}$

Above partition can be represented in bit format as

Partition 1					Partition 2					Partition 3				
a	*b*	*c*	*d*	*e*	*a*	*b*	*c*	*d*	*e*	*a*	*b*	*c*	*d*	*e*
1	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*

Fig 2 : Shows bit wise representation of a partition sample

So from figure 2 $P_1 = \{a, b, c\}$ = 11100 , $P_2 = \{e\}$ = 00001, $P_3 = \{d\}$ = 00010

3.2 Rule-Set for 3-BitPartition

- Total number of 1's in 3n bits is always equal to n.(so rest 2n bits are always zero)
- Among the three cluster, on the same bit position exactly one cluster contain value 1 rest will be zero.
  In general for n entities
  $$A_i \otimes B_i \otimes C_i = 1 \quad \text{Where} \quad 1 \le i \le n$$
  Where $\otimes$ is ex or operator.
- Each partition contains at least a single 1 and maximum n-2 1's.
- Decimal value of second partition is always less then first partition.
- Decimal value of third partition is always less then second partition.
- Decimal value of first combination (tuple) of first cluster always starts with the value of power(2,n)-4

- Decimal value of last combination (tuple) of third cluster always is power (2, n-2)-1.

Let us generate a pattern Using Rule of Belongingness(section 3.1) and using Rule-Set for 3-BitPartition(Section 3.2) for 5 entities for 3-partitioning. Generated pattern will be as shown in Table 4 (Appendix A). A unique pattern is generated for every value *n* which is different from other value of *n*. So using above rule sets 3-partitional bit patterns can be generated for any positive integer value.

3.3 Algorithm – 3-BitPartition

Input: No of entities to be partitioned (n)
Output: All possible combination of 3 partitions only in the bit pattern.

```
------------------------------------------------------------------------

3-BitPartition (int n)
    1.  Read n
    2.  Initialize i = pow(2,n) -1
    3.    for a = pow (2, n) - 4 to 1      //generate I partition.
    4.    {
    5.        j = Complement(a) & i          //generating II partition
    6.        if (j ≥ i)
    7.           break;
    8.        for b = j to 1
    9.        {
    10.          if ( no overlapping of bits between(a & b) and
                  union of (a & b) is not equal to i)
    11.          c = (complement of ( bitwise or of a & b ))
                      bitwise & with i)     // generating III partition
    12.            if ( value of third partition is less then first
                     and second partition)
    13.              {
    14.                print( dectobin(a), a, dectobin(b), b,
                          dectobin(c), c))
    15.              }
    16.          }
    17.    }

------------------------------------------------------------------------
Algorithm: Char * dectobin(int)
------------------------------------------------------------------------

Input: Takes a Decimal number.
output: A binary string

char * dectobin (int decno)
 {
    binno[0]='\0';  // binno is binary string holds binary no
    while (decno!=0)
      {
          if(decno%2==1)
            append 1 to string binno
          else
            append 1 to the string binno
          decno =decno/2;
      }
          return binno;
 }
------------------------------------------------------------------------
```

3.4 Description of Algorithm

Above 3-BitPartition algorithm generated 3 partitions in the bit format. 3- Partitions are generated in the 3n bits. Line no. 2 generates first cell (column) of partition of a particular tuple (Constraint no. 1). Line no. 7 checks complement of a is greater then a, if yes then it decrements the value of i. This step removes any repetition of tuples. In line no. 8 loop find out possible value of second cell (column). Line no 10 checks whether there is any clash of bits in first cell and in second cell (check for Rule no. 2). Line no. 10 also checks third cell should contain at least a single 1.Line no. 11 generates the third partition Line no 12 checks value of third partition is less then first and second partition. Subroutine dectobin(int) translates decimal number into a binary string which represents a partition.

3.5 Characteristics and Validation of Algorithm

Proposed algorithm is a new approach for the generation and representation of 3-Partiton. Bit approach is far better than other approach. Bit operations are used to generate partition and to check for the constraints. So reduces time complexity. In this algorithm no duplicates tuples are generated. The beauty of this algorithm is that if the first cell (column) of any tuple is not according to constraints then it does not generate second cell (column) and similarly if the generated second cell clashes with any with the first cell then it does not generate third cell (column) of the tuple. So it reduces overall running time of the algorithm.

Algorithm is validated by the number of combinations generated are equal to the Stirling number of the second kind.

This algorithm is very basic algorithm which can be applied in any of the application which requires 3-Partitioning. The most beautiful thing is that running time can be further reduced by applying Constraints. Constraints will be specific to application.

4. CONCLUSION

Bit representation of 3-Partition provides a new dimension to solve many of the partitioning algorithms. Proposed algorithm uses bit operator to generate partition. Proposed algorithm is basic 3-Paritioning algorithm which can be applied in any of the application which requires 3-Partitioning. Applying Constraints specific to application on this algorithm can further reduce the time complexity. It is the optimized partitioning algorithm as it doesn’t generates duplicate tuples and generate only 3-parititon tuples.

5. REFERENCES

[1] M.R.Garey and D.S. Johnson, Computers and Intractability: A guide to the theory of NP-Completeness, W.H. Freeman and Co., New York, 1979.

[2] Hayes Brian, “The Easiest Hard Problem”, American Scientist, May-June 2002.

[3] Zuckerman, D, “NP-complete problems have a version that's hard to approximate”, Structure in Complexity Theory Conference, 1993., Proceedings of the Eighth Annual, IEEE, 1993 , pp 305 - 312

[4] C. Ding, H. Xiaofeng, Z. Hongyuan, G. Ming, and H. Simon, “A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering,” Proc. IEEE International Conference Data Mining, pp. 107-114, 2001

[5] Y. G. Saab, “An Effective Multilevel Algorithm for Bisecting Graphs and Hypergraphs”,IEEE Transaction , June 2004, pp. 641-652

[6] M. Dell’Amico, S. Martello, “Reduction of 3-Partition Problem”, Journal of Combinatorial Optimization, 1999, Vol. 3, No. 1, 17-30

[7] R.L. Graham, D.E. Knuth, Oren Patashnik, “Concrete Mathematics-A Foundation for Computer Science”, 2nd Edition, Pearson Education

[8] J.H. Van-Lint, R.M. Wilson, “A course in Combinatorics” 2nd Edition, Cambridge University Press, 2001 pp- 119-128

[9] M. Karp, “Reducibility Among Combinatorial Problems” In R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations, 1972, pp. 85-103.