Enumeration Algorithms for Restricted and Unrestricted Compositions and Words

Daniel R. Page
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada

April 8, 2011
Last Updated: May 20, 2011

Abstract

An overview of many of the major results in enumeration algorithms for Combinatorial Compositions can be found in this survey paper. The focus of this survey paper is on unrestricted compositions, bounded compositions, and future research in restricted compositions. The paper is concerned with generating these combinatorial objects, not counting the number of element under a composition. This paper includes a brief history of the problem, and theoretical connections found between combinatorial Gray-Codes, Integer Partitions, and common techniques established over the years. Lower-bound and upper-bound time and space complexities for these class of algorithms are covered where permitted. Primary focus will be given to sequential algorithms but in certain contexts concurrent and parallel solutions may be mentioned. The motivation is to further research in w-pattern avoidance and a generalized methods in enumerating restricted compositions and other formal languages.

1 Introduction

Enumeration (Generating) algorithms have been a subject of study since the foundations of Computer Science first developed with the work of P.A.

MacMahon [1854-1929] and Axel Thue [1863-1922] in combinatorics, number theory and mathematical logic [3]. In the past decade research has grown further in interest with applications in modelling computational chemistry, bioinformatics, to cryptography and error-correcting codes.

In the past ten years enumeration (as in listing or generating) has come valuable over traditional counting algorithms because the information enumeration (listing/generating) algorithms provide are vastly superior and can represent even models. From the fields of combinatorics being a basic combinatorial object, compositions have a long history of being under intense examination and still many problems sit on some varieties of the problems with compositions. This is due to their dual nature of having an ordering and also a quantity or significance which many applications seek or desire. The motivation I had for producing this survey paper was primarily to give an overview of the problem and encourage others to further research the problem. I have a strong interest in the subject of enumeration algorithms and foundations of Computer Science so I wanted to gain more experience exploring the literature and how to properly document research. With this in mind, I myself have interest in conducting more research in these problems in terms of the research directions I provide later in the paper but are to be left to the reader as a research direction.

2 Definitions and Concepts

Counting Algorithm - A *counting* algorithm effectively computes the cardinality of a finite set given some properties of the desired construction. Returns a the intended total size of the intended construction.

Enumeration Algorithm - An *enumeration* algorithm effectively generates/lists elements of a finite set given some properties of the desired construction. Enumeration algorithms can also be augmented to solve counting problems by computing the cardindality of the generated set.

Integer Composition - A *composition* denoted by $\mathrm{C}_{k,n}$ is a set which contains any sequence $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ where $\sigma_i \in \mathbb{Z}^+$, such that $\sum_{i=1}^{n} \sigma_i = k$. Integer compositions are also referred to as *combinatorial compositions*.

Weak Integer Composition - A *weak composition* denoted by $\mathrm{C}_{k,n}$ is a set which contains any sequence $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ where $\sigma_i \in \mathbb{Z}^+ \cup \{0\}$, such that $\sum_{i=1}^{n} \sigma_i = k$. Integer compositions are also referred to as *combinatorial compositions*.

Parts - Given a sequence σ for a *composition* $\mathrm{C}_{s,n}$, the parts of a composition is denoted by $par(\sigma) = n$.

Order - Given a sequence σ for a *composition* $\mathrm{C}_{s,n}$, the order of a composition is denoted by $ord(\sigma) = s$.

Unrestricted Integer Composition - A *composition* $\mathrm{C}_{s,n}$ is *unrestricted* if and only if for all sequences $\sigma \in \mathrm{C}_{s,n}$, $ord(\sigma) \in \mathbb{Z}^+$.

Unrestricted Weak Integer Composition - A *weak composition* $\mathrm{C}_{s,n}$ is *unrestricted* if and only if for all sequences $\sigma \in \mathrm{C}_{s,n}$, $ord(\sigma) \in \mathbb{Z}^+ \cup \{0\}$.

Restricted Integer Composition - A *composition* $\mathrm{C}^R_{s,n}$ is *restricted* if and only if for all sequences $\sigma \in \mathrm{C}_{s,n}$, $ord(\sigma) \in R$ and $R \subseteq \mathbb{Z}^+$.

Restricted Weak Integer Composition - A *weak composition* $\mathrm{C}^R_{s,n}$ is *restricted* if and only if for all sequences $\sigma \in \mathrm{C}_{s,n}$, $ord(\sigma) \in R$ and $R \subseteq \mathbb{Z}^+ \cup \{0\}$.

Bounded Integer Composition - A restricted *composition* $\mathrm{C}^{L,U}_{s,n}$ is *bounded* if and only if for all sequences $\sigma \in \mathrm{C}_{s,n}$ given $L \leq U$, $ord(\sigma) \in R$ where $R = \{r \in \mathbb{Z}^+ \cup \{0\} | L \leq r \leq U\}$.

3 Formal Definitions of the Problems

In this paper we will be considering three problems. These problems are involving the generation of unrestricted compositions, bounded compositions, and restricted compositions. For future reference in this paper, we will refer to each problem by their numbering here.

Problem 1: Given $n, s \in \mathbb{Z}^+$ *(Note: if $s > 0$, then n cannot be 0)* , ef-

fectively generate the unrestricted composition $C_{s,n}$ using an *enumeration algorithm.*

Problem 2: Given $n, s \in \mathbb{Z}^+$ *(Note: if $s > 0$, then n cannot be 0)* and $L, U \in \mathbb{Z}^+ \cup \{0\}$ where L $\leq$ U, effectively generate the bounded composition $C_{s,n}^{L,U}$ using an *enumeration algorithm.*

Problem 3: Given $n, s \in \mathbb{Z}^+ \cup \{0\}$ *(Note: if $s > 0$, then n cannot be 0)* and an ordered set R where all the element in R are non-negative integers, effectively generate the restricted weak composition $C_{s,n}^{R}$ using an *enumeration algorithm.* This problem will be discussed in future research directions.

4 History of the Problem

The problems of enumerating combinatorial compositions date back as far as the late 1800's with the work of P.A. MacMahon and Axel Thue [3]. Before the days of computers, the two main concerns of compositions were concepts such as the *assemblage of objects* under a composition which were the study of MacMahon, and systematic string rewriting systems primarily by Thue. Counting for a long time was the main interest of mathematicians, but as problems grew in complexity researchers found that enumeration through generation provided a better understanding in some respects to structure [9]. From the later seventies till the late nineties there remained a large gap in the literature due to a lack of interest in generation algorithms and another rise of interest in counting algorithms due to the complexities in some counting problems. The works of Knuth, Lothaire, Ehrlich, Klingsberg, Ruskey and Stojmenovic vary greatly on the solutions to the problems of unrestricted compositions [6]. In the later half of the 20th century with the rise of computers, enumeration algorithms became widely popular in research but only till recently have really come to attention with applications across many scientific disciplines. Compositions like other basic combinatorial objects are a very important aspect to the foundations of theoretical computer science due to their relationships with word problems and are an active research problem in more specific sub-types of compositions.

5 Classical Techniques

In this section we will cover three classical techniques which still are used in practice but are not neccessarily as efficient as the modern results. These techniques can be found in *Combinatorial Algorithms: Theory and Practice* [7]. The focus of this section will be on Problem 1, that is enumeration algorithms for *unrestricted compositions*. Often these algorithms would be implemented along with using a stack, circular list, or data structure which stores distinct elements efficiently. There is a relationship between this problem and gray codes and the lower bound of Problem 1 due to it is $\Omega(k)$ due to Reingold's claims involving loopless algorithms [9, 7]. This lower bound is well established due to the relationship between Gray Codes and Unrestricted Compositions [6, 9].

Brute Force: This technique is pretty straight forward. Recursively concatenate integer sequences where the sum is less than s. Once a sequence of n parts (length n) is reached group them into a data structure for output. In very small instances this technique is efficient but can grow incredibly difficult to use in terms of time and space.

Randomization: Utilize the properties of enumerating combinations and using a disjoint data structure. Keep generating random integer sequences of length n then insert them into your circular linked list where uniqueness is required. The algorithm terminates when the size of the list is equal to $|C_{n,k}|$.

k-Bit Reflected Gray Code (Bitner-Ehrlich-Reingold Method): This result gives in the worst case O(1 + k) for only very particular Gray codes and is a loopless algorithm [7]. This result has been updated in 2003 [9]. The idea of the algorithm is to utilize the exhaustion of combinations by concatenating bits at the bit level to form integer sequences of a fixed size. Often a stack is used in implementation. Once the length n of bits are formed, a single pass is made eliminating elements in the list whose sum are not s (through pruning).

6 Modern Results

We will consider modern results to begin at 1999 following the development of the ECO Method for enumeration of combinatorial objects. Between the late 1970's and the late 1990's the literature on generation algorithm was pretty sparse following the developments of Knuth, Lothaire and Reingold among others. Once the ECO methodology was developed, there was a growing interest in combinatorial objects once again with the further access of computational power. In turn many applications from genetics, to digital transmissions as a standard on the rise so interest in these enumeration algorithms arose once again.

As I mentioned in classical techniques, there is a $\Omega(k)$ lower bound on Problem 1. The algorithms in the classical techniques could not reach that optimal goal but some results in a related discourse would come very much in handy.

In 2003, Timothy Walsh developed an algorithm which can generate Gray Codes in O(1) worse-case time for almost all gray codes [9]. What makes this result different than that the Bitner-Ehrlich-Reingold Method devised in classical techniques which is also a *loopless* algorithm which has O(1) worse case-time (only for the computation of the gray codes), is that the B.E.R. method only works on very specific Gray Codes which really limits it's purposes and in practice would increase the running time. Walsh's paper establishes that Gray codes can infact be used in a practical manner by generalizing all the main techniques of representing Gray Codes [9].

Now with this result in hand, Problem 1's solution is optimal by augmenting Walsh's algorithm with the B.E.R. method such that we obtain $\Theta(k)$-worse case running time, where k = $|C_{s,n}|$.

Following this result in 2003, many techniques would follow adapting Walsh's algorithm to the ECO method to generate gray codes from the classical techniques we have covered in the previous section. A good example is in 2004, the work done by Silvia Bacchelli, Elena Barcucci, Elisabetta Grazzini, and Elisa Pergola on exhaustive generation algorithms for gray code structures (with a main focus on combinations, permutations, and subsets) [1]. To remain on topic, in 2007 Antonio Bernini, Elisabetta Grazzini, Elisa Pergola, and Renzo Pinzani rediscovered Walsh's algorithm but in the context

of developing Gray Code structures which in turn offer more of the classical options for Problem 1 to be applied once more [2]. Lastly, in 2008 Walsh's technique was extended by Toufik Mansour, and Ghalib Nassar by improving the *hamming distance* of Knuth's and Ruskey's Gray Codes [5].

In 2006, a brief parallel solution for Problem 1 was found to run in O(C(m,n) log m + k) that uses the combination enumeration technique, where m is the number of processors, C(m,n) is m choose n, n is the number of parts, and k is the number of elements in that unrestricted composition [8].

With Problem 1 having an optimal solution now, some researchers in theoretical computer science and combinatorics became interested in generalizing particular types of *restricted compositions*. The idea was that in some cases one would not want all the listings but only a subset of them which contained very specific elements over a continuum. Say if one were analyzing a statistical set of data or only wanted a genomic sequence of 4 symbols instead of exhausting out numerous which don't even have 4 symbols [6, 3]. These would fall under Problem 2 which involve *bounded compositions*. It is trivial to see that one could apply similar techniques to the same problem but more checking would need to occur for which elements belong in the list or not which can be very costly if the sequences are long. In most instances, you would also be wasting a significant amount of space storing the computation in contrast to the actual solution so researchers tried looking for new techniques. Some would look at bounded compositions from only the upper end, modifying the values of U and fixing L=0, or L=1 [6].

Previous to Walsh's result Clark Kimberling developed a technique for counting compositions using Pascal's Triangle and Fibonnaci numbers. The result allows one to form paths in the interiors of Pascal's triangle off of *off diagonals* that run up recursively in the triangle to the top to obtain the different partitions of the composition. That is, just permuting the elements after obtaining each off diagonal sum (these off diagonals are infact compositions) could be combined with rows above them if a sum is not met. Kimberling did not intend this for generating but the counting method stuck. This result by Kimberling was discovered in 2002 [4]. Not till 2010 did someone make use of this technique to realise it could be used for generation aswell. Ends up a solution for Problem 2 derived from this technique by J.D. Opdyke through his RICs algorithm for generating bounded integer compositions [6].

Opdyke's solution recursively will traverse Pascal's triangle entries and then add it to a previous row if a sum was not met (much like you would see in a Fibonnaci sequence in essence). The running time of the algorithm is not tightly bounded because there does not exist yet a counting closed form for bounded compositions according to Kimberling [6, 4]. The running time given in the paper is $\sim O(s)$ per composition so this form essentially has complexity $O(kR)$ where R is the number of bounded compositions [6]. This is the most general solution from these class of problems but it inheritely still contains flaws which are address in the paper [6]. This is a solution to Problem 2.

7 Issues in Algorithms for Modern Results in Sequential Algorithms

The issues I would like to discuss are with the current solution for Problem 2. The solution we currently have does not cover the entire domain of possible choices for symbols in the non negative integer sequences. That is, the RICs algorithm has several flaws compared to the solution for Problem 1 which could maybe be improved. The concept of bounded compositions that are *doubly bounded* like what someone may consider for Problem 3 may just be an algorithm which can't be loop-free sequentially.

Firstmost, unlike the solution to the first solution, this algorithm uses loops and recursion which can add a heavy cost in computation for large instances of the bounded compositions but is quite efficient for small instances. Secondly, the domain of potential values for L and U are not covered in the RICs algorithm. L or U cannot be 0 in Opdyke's algorithm because the algorithm is defined where weak integer compositions are not allowed (the element 0 is *forbidden* in non-weak integer compositions) [6]. Often some sources will rule out the weak compositions because their complexities due to the empty cases, yet being the general case [3]. More research may be required to understand the trade-offs between increasing the complexity of compositions and their algorithmic structure in theory.

8 Future Research Directions

Pattern Avoidance: Currently there is a strong interest in pattern avoidance in strings. To begin let us define what pattern avoidance is in terms of enumeration because counting results are different. A $w-pattern\ avoidance$ on compositions or words is where we wish to generate a type of composition which *avoids* patterns which is a potential w in the non-negative integer sequences found in the composition C [3]. In contrast than traditional conventions, unlike looking for frequencies of common subsequences over a composition, this is the opposite.

There exist some results currently present in the literature over subsets of this problem. Over subsequent patterns of length three, Mansour and Vainshtein first developed a method called the *block decomposition* method for enumeration [3]. Combinatorists share a strong interest in counting these number of avoidances but they are rather difficult to enumerate so *enumeration algorithms* may be an approach to take with these problems where generating functions may not be found. Generating compositions which avoid particular strings can have evident applications where we wish to eliminate particular subsequences from a listing. I would suspect such would be of interest for applications for particular constructions in coding theory, or statistical analysis of strings of data.

Optimization of Problem 2: It is evident that we have not obtained the optimal solution for Problem 2, or if there exists a tighter lower bound for bounded compositions. The run-time we have currently for this algorithm from Opdyke is $\sim O(s)$ per composition, where s is the order [6]. That could be proving a tighter lower-bound or finding a better upper bound for this problem. The goal would be an even further improved version of Opdyke's RICs enumeration algorithm.

Problem 3 and further Generalization: This problem in itself has a strong relationship with the decision problem of positive integer Subset-Sum except where we consider target $t = s$ where s is the order of the weak composition. Essentially Problem 3 is the answer to all possible solutions for such a query of non-negative integer solutions (for weak compositions). Subset-Sum is an NP-complete problem. The primary difference is that we do not acknowledge the concept of maximizing t but only keeping sequences which

sum to t so it is not exactly the NP-complete problem but it is suspected to have many intractable properties that can be explored.

It would be of great interest in many applications to produce a purely generalized method for handling any restricted weak composition generation request. This may be constructing the algorithm or extending the algorithm to produce equivalent requests or generalizing the problem even further. It is expected not to obtain optimal times with such a solution but a feasible usable solution that solves the problem.

This could be even extending or meshing this with pattern avoidance per symbol. That is, each symbol contains a restricted set R much like Problem 3 states for an entire sequence.

Compositions and Partitions: J.D. Opdyke concludes his paper with suggesting more research be done into the counting properties between restricted partitions and restricted compositions through enumeration. In Opdyke's paper he finds an interrelationship between the *algorithmic* properties between partitions and compositions so it merits some research into an *algorithmic* link between the two objects further [6].

9 Conclusions

Based on the current research found in the problems beyond the ones stated, there is much to explore in terms of combinatorial algorithms between combinatorial objects such as *Combinatorial Compositions*. I personally found in gathering research between fields rather difficult due to the language used in specific fields. Some experts use the term enumeration for counting, while others use enumeration as a term for generation. I also wanted to cover more parallel and concurrent solutions but it appeared that the problems out there currently had more than enough interesting results already found. Since my primary interest was sequential algorithms, these other types of solutions became less valuable for this survey of the literature. Overall, I wish to conclude that enumeration algorithms for restricted compositions merit more research because they have valuable applications as mentioned and have theoretical significance. I wish the best for any further research on the topic.

References

[1] S. Bacchelli, E. Barcucci, E. Grazzini, and E. Pergola. Exhaustive generation of combinatorial objects by eco. *Acta Informatica*, (40):585–602, 2004.

[2] A. Bernini, E. Grazzini, E. Pergola, and R. Pinzani. A general exhaustive generation algorithm for gray structures. *Acta Informatica*, (44):361–376, 2007.

[3] S. Heubach and T. Mansour. *Combinatorics of Compositions and Words*. CRC Press, 2010.

[4] C. Kimberling. Path-counting and fibonacci numbers. *Fibonacci Quarterly*, 40(4):328–338, 2002.

[5] T. Mansour and G. Nassar. Gray codes, loopless algorithm and partitions. *Journal of Mathematical Modelling and Algorithms*, (7):291–310, 2008.

[6] J. Opdyke. A unified approach to algorithms generating unrestricted and restricted integer compositions and integer partitions. *Journal of Mathematical Modelling and Algorithms*, 9(1):53–97, 2010.

[7] E. M. Reingold, J. Nievergelt, and N. Deo. *Combinatorial Algorithms: Theory and Practice*. Prentice-Hall Inc., 1977.

[8] I. Stojmenovic. Listing combinatorial objects in parallel. *The International Journal of Parallel, Emergent and Distributed Systems,*, 21(2):127–146, 2006.

[9] T. Walsh. Generating gray codes in o(1) worse-case time per word. *DISCRETE MATHEMATICS AND THEORETICAL COMPUTER SCIENCE, PROCEEDINGS*, 2731:73–88, 2003.