CSCanada
Progress in Applied Mathematics
Vol. 2, No. 1, 2011, pp. 61-66
DOI: 10.3968/j.pam.1925252820110201.010

ISSN 1925-251X [Print]
ISSN 1925-2528 [Online]
www.cscanada.net
www.cscanada.org

Generation of All Possible Multiselections from a Multiset

Thomas Wieder[1,*]

[1]Germany
[*]Corresponding author.
Address: Germany. Email: thomas.wieder@t-online.de

Received 6 May 2011; accepted 16 May 2011

Abstract
The concept of a $[k_1, k_2, \ldots, k_K]$-selection applied on a multiset is introduced and an algorithm is outlined to generate all $[k_1, k_2, \ldots, k_K]$-selections from a given multiset.

Key words
Multiselection; Multisets; Contingency matrix; Combinatorics

Thomas Wieder (2011). Generation of All Possible Multiselections from a Multiset. *Progress in Applied Mathematics, 2*(1), 61-66. Available from: URL: http://www.cscanada.net/index.php/pam/article/view/j.pam.1925252820110201.010
DOI: http://dx.doi.org/10.3968/j.pam.1925252820110201.010

INTRODUCTION

We will introduce (quite informally) the notion of a multiselection and apply it to a multiset. Then we will outline an algorithm which generates all possible multiselections for a given multiset. A corresponding Maple program named *Multiselection.mpl* will be described. The enumeration of all possible multiselections for a given multiset will be mentioned. Finally, we will give possible applications and we will discuss possible amendments of *Multiselection.mpl*.

1. MULTISELECTIONS FROM A MULTISET

Consider a multiset $\mathbf{N} = [e_1, \ldots, e_1, e_2, \ldots, e_2, \ldots, e_j, \ldots, e_N]$ composed of N elements $e_1, e_2, \ldots, e_N$ where each element e_j occurs with the multiplicity m_j in $\mathbf{N}$. Thus, in a multiset an element e_j may occur several times in contrast to a usual set. As a consequence, the multiple instances of e_j can not be discerned as it is in the case of a set of unlabeled elements. However, different elements e_j and $e_{j'}$ can be discerned as it is in the case of a set of labeled elements. Order does not count within $\mathbf{N}$. [2] An equivalent way to write

[2] In order to handle with a multiset in a program, it is useful to introduce an ordering like the lexicographic ordering but this ordering does not enter into enumeration questions.

N uses the multiplicities in the obvious manner $\mathbf{N} = [(m_1, e_1), (m_2, e_2), \ldots, (m_N, e_N)]$. As an example, the multiset $\mathcal{N} = [a, a, a, b, b, b, b, c] = [(3, a), (4, b), (1, c)]$ is composed of the three elements a, b and c with multiplicities $m_1 = 3$, $m_2 = 4$, $m_3 = 1$ and what regards order, $[b, b, a, c, a, b, a, a, b]$ is the same multiset.

Now consider the task to select k (not necessarily distinct) elements from **N** with $k \leq N_t$ where $N_t = \sum_{j=1}^{N} m_j$. We will speak of a k-selection. Then the question arises how many k-selections are possible for given **N**. We will abbreviate the number of possible selections by Z.[3] MacMahon gave the quite complicated formula for Z in the case of a multiset $\mathcal{N}$, see appendix.

Finally we arrive at the last step of our problem formulation. We want to perform a selection which is prescribed by a multiset $\mathcal{K} = [k_1, k_2, \ldots, k_K]$ of K selection numbers k_i. We will speak of a $[k_1, k_2, \ldots, k_K]$-selection-list or just of a selection list. The sum $\sum_{i=1}^{K} = \kappa$ is the total number of selected elements. We discard any order among the k_i in $\mathcal{K}$. Thus, our selection $\mathcal{S}$ will be a set of subsets $[\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_i, \ldots, \mathcal{S}_K]$ where the i-th subset $\mathcal{S}_i$ will have k_i elements taken from $\mathcal{N}$. Obviously, $\mathcal{S}_i$ may be a multiset itself. Because of this set of subsets, we speak of a multiselection instead of a selection only.

2. A NON-RECURSIVE ALGORITHM VIA SELECTION MATRIX

If we are given with a multiset $\mathcal{N}$ and a $[k_1, k_2, \ldots, k_K]$-selection-list, then Z possible multiselections $\mathcal{S}$ exist. We note that Z does not depend on the order of the selection numbers k_i. No formula for Z is known yet. [4]

We are ready to describe our algorithm to generate all these multiselections $\mathcal{S}$. The problem is to keep track of the already selected instances of each multiple element. The solution consists in the usage of a selection matrix **X** with its elements $x_{i,j}$ where $i = 1, \ldots, K$ and $j = 1, \ldots, N$. Row i of **X** corresponds to the selection element k_i. Column j of **X** corresponds to the multiset element e_j. Equation (1) depicts these correspondences, the row sums are $\sum_{j=1}^{N} x_{i,j} = k_i$ and the column sums are $\sum_{i=1}^{k} x_{i,j} = w_j$.

$$\mathbf{X} = \begin{array}{cccc|c} e_1 & e_2 & \ldots & e_N & \\ \hline x_{1,1} & x_{1,2} & \ldots & x_{1,n} & = k_1 \\ x_{2,1} & x_{2,2} & \ldots & x_{2,n} & = k_2 \\ \ldots & \ldots & \ldots & \ldots & = \ldots \\ x_{3,1} & x_{3,2} & \ldots & x_{3,n} & = k_L \\ \hline w_1 \leq m_1 & w_2 \leq m_2 & \ldots & w_3 \leq m_N & \end{array} \tag{1}$$

As an example for **X**, we return to the multiset $\mathbf{N} = [a, a, a, b, b, b, b, c]$ and request the multiselection $\mathcal{K} = [2, 3, 1]$ to be taken from **N**. Several solutions are possible, one of them is outlined in the following selection matrix:

$$\mathbf{X} = \begin{array}{ccc|c} e_1 = a & e_2 = b & e_3 = c & \\ \hline 1 & 1 & 0 & 2 = k_1 \\ 1 & 2 & 0 & 3 = k_2 \\ 0 & 0 & 1 & 1 = k_3 \\ \hline w_1 = 2 & w_2 = 3 & w_3 = 1 & \end{array} \tag{2}$$

From (4) we see that our particular solution is $\mathcal{S} = [[a, b], [a, b, b], [c]]$. However, also $\mathcal{S}' = [[a, a, a], [b, b], [c]] = [[b, b], [a, a, a], [c]]$ is a valid solution (remember that the order of the subsets $\mathcal{S}_i$ does not count), but this solution would have a different selection matrix.

[3] If **N** is a usual set of N distinct elements, then the answer is the binomial coefficient $Z = \binom{n}{k}$ and for N_t identical elements the answer is $Z = 1$.

[4] Z is easy to calculate for a conventional set with N_t distinct elements, namely $Z = \binom{N_t}{k_1}\binom{N_t-k_1}{k_2}\ldots\binom{N_t-k_1-\ldots-k_{K-1}}{k_K}$. As above, if all elements are identical, then $Z = 1$.

The selection matrix **X** shows us a way to systematically generate all possible solutions. The sum over all row sums is $\sum_{i=1}^{K} k_i = \kappa$, that is the total number of selected elements. The column sum w_j is the number of chosen instances of element e_j and it must be fulfilled that $w_j \leq m_j$. Finally, the sum $\sum_{j=1}^{N} w_j$ over all column sums must be equal to κ, too. Thus, **X** is a matrix with given row sums and the column sums are restricted to the condition $\sum_{j=1}^{N} w_j = \kappa$.

To each feasible solution one and only one selection matrix corresponds. In order to find all feasible solutions, we have to construct all corresponding selection matrices. This is done by finding all N-tuples of integers with $\sum_{j=1}^{N} w_j = \kappa$, each of these tuples is an allowed column sum vector. Lower and upper limits apply for the tuples: The lower column marginals are equal to zero for each element e_i and the upper marginals are equal to the multiplicities m_i. The row sum vector is always the same, namely $k_1, k_2, \ldots, k_K$. Thus, for each of these N-tuples we are confronted with a matrix **X** with given row and column sums and our task is then to find all possible realizations of **X**. This problem is well know in statistics, it is the generation of all $r \times c$-tables or also the generation of all contingency matrices with given row and column sums [1, 2].

We can write down this ansatz quite formally. Let us introduce the following symbol for the two positive integers m and n.

$$\left| \begin{array}{c} m \\ n \end{array} \right| = \begin{cases} 1 \text{ if } n \leq m \\ 0 \text{ else} \end{cases} \tag{3}$$

Furthermore, let us understand that in (4) a sum like $w_{i,1} + \ldots + w_{i,j} + \ldots + w_{i,N} = k_i$ indicates all integer partitions of K_i with $w_{i,j} \leq m_j$. The array of partitions under the summation shall mean that we have to count all feasible combinations of the K partitions corresponding to the K subsets of the $\mathcal{K}$-selection. In (4) the product guaranties that only those partitions contribute to Z for which the number of used instances of element e_j is less or at most equal to its multiplicity m_j. Then the number $Z = Z(\mathcal{K})$ of all possible $\mathcal{K}$-selections from the multiset $\mathcal{N}$ is

$$Z(\mathcal{K}) = \sum_{\substack{w_{1,1} + \ldots + w_{1,j} + \ldots + w_{1,N} = k_1 \\ \ldots \\ w_{i,1} + \ldots + w_{i,j} + \ldots + w_{i,N} = k_i \\ \ldots \\ w_{K,1} + \ldots + w_{K,j} + \ldots + w_{K,N} = k_K}} \prod_{j=1}^{N} \left| \begin{array}{c} m_j \\ w_{1,j} + \ldots + w_{i,j} + \ldots + w_{K,j} \end{array} \right| \tag{4}$$

An algorithm for the generation of all integer N-tuples with given lower and upper bounds has been provided by O'Connor [3]. An algorithm for finding all $r \times c$-tables was given by Saunders [4]. Then our algorithm for generating all multiselections for the given selection list $\mathcal{K}$ from the multiset $\mathcal{N}$ proceeds as follows:

1. Generate all feasible N-tuples.
2. For each N-tuple generate all corresponding selection matrices.
3. Transform each selection matrix into a selection list $\mathcal{S}$.

This algorithm has been implemented into a corresponding Maple (version 13) program named *Multiselection.mpl* [5]. It takes two arguments, $\mathcal{K}$ and $\mathcal{N}$. As an example of its use, we take $\mathcal{K} = [2, 3, 1]$ and $\mathcal{N} = [a, a, b, b, b, c]$ (where we actually use Maple's list construction instead of Maple's multiset construction). Then the command *Multiselection*([2, 3, 1], [a, a, b, b, b, c]) gives us [[[c,a], [a,b,b], [b]], [[c,a], [b,b,b], [a]], [[c,b], [a,a,b], [b]], [[c,b], [a,b,b], [a]], [[a,a], [c,b,b], [b]], [[a,a], [b,b,b], [c]], [[a,b], [c,a,b], [b]], [[a,b], [c,b,b], [a]], [[a,b], [a,b,b], [c]], [[b,b], [c,a,a], [b]], [[b,b], [c,a,b], [a]], [[b,b], [a,a,b], [c]]]. [5]

[5] Actually, *Multiselection.mpl* returns two lists where the first list is an intermediate result and the second list is the final result. Our algorithm produces ordered selections first, then multiple selections are removed. For example, with $\mathcal{K} = [2, 1, 2]$ and $\mathcal{N} = [a, a, b, b, b, c]$ we get 24 selections in the intermediate result. Except of [[a, b], [b], [a, b]] and [[a, b], [c], [a, b]], all other solutions

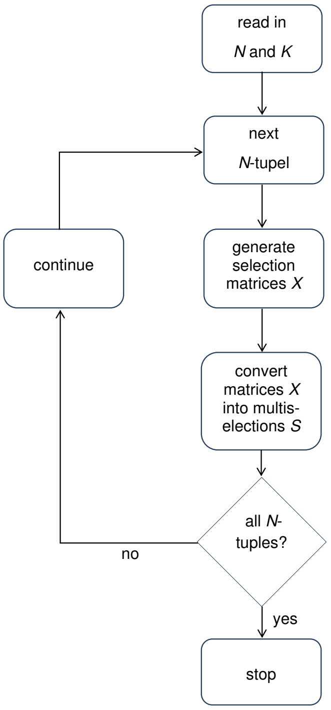

Figure 1
Program Flow for Multiselection. mpl.

3. APPLICATIONS

The first application is the complete set partitioning of $\mathcal{N}$. Consider an integer partition $\mathcal{P}$ of the total number of multiple elements $N_t = \sum_{j=1}^{N}$ in $\mathcal{N}$. We understand $\mathcal{P}$ as a selection list (like $\mathcal{K}$ above) and we generate all corresponding multiselections from $\mathcal{N}$. Let us call the set of all these partitions $\mathcal{P}(N_t)$. If we perform all associated multtiselections for $\mathcal{P}(N_t)$, then we arrive at the complete set partitioning $\Sigma_{\mathcal{N}}$ of $\mathcal{N}$. As an example, consider the multiset $\mathcal{N} = [a, a, a, b, c]$. Since $N_t = 5$ we have $\mathcal{P}(N_t) =$ [[1, 1, 1, 1, 1], [1, 1, 1, 2], [1, 2, 2], [1, 1, 3], [2, 3], [1, 4], [5]] and we get $\Sigma_{\mathcal{N}} =$ [[[a], [a], [a], [b], [c]], [[a], [a], [a], [b, c]], [[a], [a], [b], [a, c]], [[a], [a], [c], [a, b]], [[a], [b], [c], [a, a]], [[a], [a, a], [b, c]], [[a], [a, b], [a, c]], [[b], [a, a], [a, c]], [[c], [a, a], [a, b]], [[a], [a], [a, b, c]], [[a], [b], [a, a, c]], [[a], [c], [a, a, b]], [[b], [c], [a, a, a]], [[a, a], [a, b, c]], [[a, b], [a, a, c]], [[a, c], [a, a, b]], [[b, c], [a, a, a]], [[a], [a, a, b, c]], [[b], [a, a, a, c]], [[c], [a, a, b]], [a, a, a, b, c]].

Furthermore, several combinatorial objects can be constructed using a multiselection on a suitable $\mathcal{N}$. One example are the so-called "hierarchical orderings" [6]. Multisets are also used in cluster analysis where optimal clustering of objects is the aim [7]. If we have to distribute elements into a multiset in an optimal manner, then (in principle) we need to produce all possible combinations and its here where multichoose could come into play. A common example is the distribution of goods among recipients where we may ask for the most fair distribution. Finally we want to mention that counting of multisets of a certain structure results in many well-known (and sometimes in unknown) integer sequences. For example, the examina-

occur twice, e.g. [[c, a], [a], [b, b]] = [[b, b], [a], [c, a]]. We leave this first list in the output because it corresponds to the case when order of subsets in $\mathcal{S}$ counts. In the second list returned by *Multiselection.mpl* all multiple instances are removed. In the above example we thus get 13 solutions only: [[[b], [a, b], [a, b]], [[c], [a, b], [a, b]], [[a], [a, b], [b, b]], [[a], [a, b], [c, b]], [[a], [b, b], [c, a]], [[a], [b, b], [c, b]], [[b], [a, a], [b, b]], [[b], [a, a], [c, b]], [[b], [a, b], [c, a]], [[b], [a, b], [c, b]], [[b], [b, b], [c, a]], [[c], [a, a], [b, b]], [[c], [a, b], [b, b]]].

tion of [1,1]- and [2,2]-multiselections applied to the natural numbers (A000027) and the natural numbers repeated (A008619) resulted in findings which are listed under the sequence entry A188667 in the OEIS [8].

4. DISCUSSION

The literature on multisets is not vast. An introduction into the combinatorics of multisets was given by Petrovsky [11]. Various applications of multisets were outlined by D. Singh et al. [12], the connection of multisets to membrane computing and DNA computing was described by G. Ciobanu and M. Gontinea [13]. In particular, only a few papers on the enumeration or generation of submultisets of multisets exist. Hage gave a formula for the particular case in which each element has equal multiplicity [14]. Furthermore, Ruskey and Savage provided a Gray code for k-combinations of a multiset [15]. To the author's best knowledge, no literature exists on multiselections from multisets.

According to an extensive literature search, our algorithm seems to be the first one form the generation of multiselections from multisets. However, the present algorithm has two drawbacks.
1.: *Multiselection.mpl* returns all its solution at once, for large multisets it would be better to get one solution at a time.
2.: At first, *Multiselection.mpl* produces ordered multiselections from which the (unordered) multiselections (i.e. without permutations of the subselections) are sorted out, see the corresponding footnote above. This is inefficient.
The reason for this detour lies in subroutine *enum.mpl* which needs to be replaced by more efficient version. Aside from these deficiencies the algorithm is non-recursive which makes it more easy to understand during execution compared to a recursive algorithm. A recursive algorithm was developed by the present author previously [16]. This algorithm inevitably results in ordered multiselections.

Thus, a more efficient non-recursive algorithm is left to future work. The same applies to the derivation of a closed-formed expression for the number of $\mathcal{K}$-selections from multisets.

ACKNOWLEDGEMENT

Many thanks to Joe Riel, Maplesoft, for a valuable discussion on multiselections from multisets.

APPENDIX

MacMahon's formula is [9]

$$Z = \sum_{t=0}^{N_t} (-1)^t \sum_{1 \le i_1 < i_2 < \ldots < i_t \le N_t} \binom{N_t + k - m_{i_1} - m_{i_2} - \ldots - m_{i_t} - t - 1}{N_t - 1}$$

The second sum in (5) runs over all possible subsets of size t which can be taken from the multiplicities. For example, if $\mathbf{N} = [a, a, a, b, b, b, b, c]$, then one can get for $k = 5$ the following 6 selections [a, a, a, b, b, b], [a, a, a,b, b, c], [a, a, b, b, b, b], [a, a, b, b, b, c], [a, b, b, b, b, c] [16].

REFERENCES

[1] Diaconis, P., & Gangolli, A. (1995). Rectengular Arrays with Fixed Margins. In D. Aldous (Ed.), *Discrete Probability and Algorithms* (pp. 15-41). New York: Springer Verlag.

[2] Greselin, F. (2003). Counting and Enumerating Frequency Tables with Given Margins. *Statistica & Applicazioni, 1*(2), 87-104.

[3] O'Connor, D. (2006). A Minimum Change Algorithm for Generating Lattice Points. Retrieved from http://www.derekroconnor.net.

[4] Saunders, I. (1984). Algorithm AS 205, Enumeration of R x C Tables with Repeated Row Totals. *Journal of the Royal Statistical Society: Series C (Applied Statistics), 33*(3), 340-352.

[5] *Multiselection.mpl* can be found at http://thomas-wieder.privat.t-online.de/Multiselection.mpl. The corresponding Maple 13 worksheet is available at http://www.thomas-wieder.privat.t-online.de/Multiselection.mw.

[6] Sloane, N. J. A., & Wieder, T. (2004). The Number of Hierarchical Orderings. *Order, 21*(1), 83-89.

[7] Petrovsky, A.B. (2003). Cluster Analysis in Multiset Spaces. *Lecture Notes in Informatics, P-30*, 109-119.

[8] Encyclopedia of Integer Sequence (2011). Retrieved from http://oeis.org/A188667.

[9] The formula is attributed to Percy Alexander MacMahon, but I failed to find it in his Combinatory Analysis and in his Collected Papers.

[10] A Maple program which implements the formula (5) can be found at http://thomas-wieder.privat.t-online.de/MacMahonsMultisetFormula.mpl.

[11] Petrovsky, A.B. (2000). Combinatorics of Multisets. *Doklady Mathematics, 61*(1), 151-154.

[12] Singh, D., Ibrahim, A.M., Yohanna, T., & Singh, J.N. (2007). An Overview Of the Application of Multisets. *Novi Sad Journal of Mathematics, 37*, 73-92.

[13] Ciobanu, G., & Gontinea, M. (2009). Encodings of Multisets. *International Journal of Foundations of Computer Science, 20*(3), 381-393.

[14] Hage, J. (2003). Enumerating Submultisets of Multisets. *Information Processing Letters, 85*, 221-226.

[15] Ruskey, F., & Savage, C. (1996). A Gray Code for the Combinations of a Multiset. *European Journal of Combinatorics, 17*, 493-500.

[16] Unpublished, the corresponding Maple 13 worksheet including source code is available at http://www.thomas-wieder.privat.t-online.de/multichoose.mw.