1979

# Rapid calculation of elemental compositions for high resolution mass spectra data

R. Geoff Dromey
*University of Wollongong*

Gordon T. Foyster
*University of Wollongong*

# RAPID CALCULATION OF ELEMENTAL COMPOSITIONS FOR HIGH RESOLUTION MASS SPECTRAL DATA

R. Geoff Dromey*and Gordon T. Foyster

Department of Computing Science and Computer Centre, University of Wollongong, P.O. Box 1144, Wollongong, N.S.W. 2500, Australia.

Abstract:

When the calculation of elemental compositions for high resolution mass spectral data is structured in such a way as to minimize the number of steps to generate each new candidate considerable gains in efficiency can be achieved. Furthermore, if meticulous care is taken to only examine those regions of the composition search space that can possibly lead to valid compositions then even much larger gains in computational efficiency can be made. Both these aspects of the elemental composition calculation are explored in detail. The outcome has been the development of a new algorithm for elemental composition calculations that is approximately 100 times faster than currently available algorithms for typical spectra. The new algorithm alleviates the problem of excessive computation times for both high mass values and for situations where six or more element types must be considered.

*Author to whom correspondence should be addressed.

Brief:

A highly efficient algorithm for generating elemental compositions is presented. The method uses advanced techniques for structuring the calculation so as to achieve approximately a 100-fold gain in efficiency over existing algorithms for typical data.

## INTRODUCTION

The assignment of possible elemental compositions to peaks in a high resolution mass spectrum has proved to be a formidable computation even for modern high-speed computers. The combinatorial nature of the problem almost invariably makes the calculation a time-consuming task. The development of computerized high resolution mass spectrometer (HRMS) systems has resulted in a greatly enhanced capacity for generating high resolution data. This is particularly so for GC-HRMS-computer systems.

These developments together with the increasing employment of high resolution mass spectrometry in analytical applications have served to underline the need for very efficient computation of elemental compositions.

A number of computer algorithms (1-3) and methods (4,5) have been suggested for the computation of elemental compositions. These algorithms perform satisfactorily when only 3 or 4 element types are considered. Beyond this they are usually unacceptably slow. Robertson and Hamming (1) in their treatment of the problem have recognized the difficulty of performing these computations rapidly for other than a limited set of elements with very restricted ranges. By use of a relatively sophisticated backtrack programming technique they were able to develop an algorithm that was significantly more efficient and more widely applicable than earlier schemes.

In the present work it will be shown how to make close to a further 100-fold gain in computational efficiency.

## BASIS OF ALGORITHM

To place the current work into perspective the problem shall be defined more explicitly and it will be shown how earlier algorithms were used to solve it.

In computing the elemental composition for any particular mass what must be done is first to decide on what elements may be present in the composition. The problem must then be further constrained by setting limits on the number of atoms of each type that could realistically (chemically) be present in the composition (e.g. the upper limits might be for example, $C_{24}$ $H_{50}$ $O_8$ $N_8$ $Cl_4$ $S_4$). The task then is to find <u>all</u> combinations of elements from these ranges that add up to the accurate mass being considered (that is, to within some predefined error tolerance, perhaps 5 ppm). A little thought reveals that this amounts to 24x50x8x8x4x4 possible combinations that need to be examined.

In any given problem only a minute fraction of these combinations are successful in satisfying the constraints imposed by the accurate mass and its associated error tolerance. Earlier algorithms for solving this problem did so by essentially generating all possible combinations (perhaps with the exception of carbon which was factored out) and then checking each combination against the accurate mass. The reason such an approach is so slow is because of the large number of candidates that are generated which have *no chance* of leading to a feasible solution. Clearly therefore it is necessary to look for *criteria* and *mechanisms* that will very drastically reduce the number of candidates generated in order to significantly improve the efficiency of this computation.

(a) Mechanism for Composition Generation

With regard to mechanism it can be seen that for each combination generated the *sum* of atomic weights of all elements present in the composition must be made. If this involves p element types then p-1 additions will be needed (e.g. $C_8 + H_{12} + O_4 + Br_2$ might be one combination generated). Fortunately it is not necessary to perform such a large number of additions to generate each *new* combination. In fact the computation can be structured in such a way that *only one* addition needs to be associated with each new combination generated.

To achieve this minimization in computations for generating each new combination a branch and bound algorithm (6) is used which incorporates a stack mechanism. The latter method although similar in principle to Robertson and Hamming's algorithm involves a simpler implementation. This is borne out by the fact that the computation time per formula tested (see tables I and III) is an order of magnitude less than for Robertson and Hamming's algorithm (1).

A simple example best describes how the mechanism operates. Consider the problem of generating all valid combinations for the elements O, Br and N which have upper limits of 8, 4 and 8 respectively. Also assume that the mass for which the elemental composition is to be derived is 160 (for simplicity of presentation all mass defects have been ignored - this *does not* happen in the actual computation).

The stepwise path of the computation is illustrated in figure (1) and figure (2). The algorithm starts with zero contributions from all elements (e.g. $O_\emptyset$ $Br_\emptyset$ $N_\emptyset$) the mass stack is zero, and the balance deficit (which is to be made up by carbons and hydrogens) is equal to the mass (160) whose composition is being sought. For this "zero" configuration a composition consisting only of carbons and hydrogens is generated. The next composition is generated by adding one nitrogen, this increases the *mass stack* to 14 and correspondingly decreases the

*balance* deficit by 14 to yield 146. The carbon and hydrogen combination that yields the mass 146 is then generated and the mass of the completed composition is tested against the starting mass. A second nitrogen is then added to the composition, the mass stack increases to 28, and the balance deficit decreases to 132. Again the carbon-hydrogen computation and the error checks are made.

The process continues until the number of nitrogens is exhausted or until the balance deficit goes negative (in this example the balance deficit is still positive when the maximum number of nitrogens have been added). If this happens, the nitrogen contribution to the balance deficit and the mass stack is removed and a single contribution from the next element is added to the mass stack. The process then continues by adding nitrogens one by one as indicated in figures (1) and figures (2). As soon as the mass stack overflows the nitrogen contribution is zeroed, the bromine contribution is increased by one, and then the nitrogen addition starts again. When the Br contribution overflows with a zero nitrogen contribution, it too is reset to zero. The next element oxygen is then incremented by 1 and so the whole pattern repeats. A detailed flowchart for the algorithm is given in figure (3). The dynamic limiting of the number of contributions from each element is more effective than the static limiting method of Burlingame (2).

In implementing this algorithm the order in which successive elements are included in the composition calculation is important. *To limit the tree-like growth of the search space examined by the algorithm the elements that have the widest ranges should be included last* (e.g. if up to 4 bromines are possible and up to 8 nitrogens are allowed then the nitrogens should be included later). In general to discover a composition that cannot lead to a possible solution several elements usually need to be included. It follows that if the major part of the

fan-out of the combinatorial space occurs well away from the root of the tree then more compositions can be precluded using the order suggested. This pruning of the search space is reinforced by fact that the elements with the largest masses generally require the smallest ranges.

There is yet one other refinement for improving the present algorithm. It involves storing partial compositions (or subtrees) as they are generated. This saves recalculation of subtrees when they are needed at different times during the computation. This can lead to considerable savings in the computation time (6). It has not however been implemented because it is usually rather costly in the amount of storage that is required. Where higher efficiencies are necessary this approach should be taken although it is necessary to pay the price of much higher storage costs. There are still other very substantial gains in efficiency that can be made over and beyond the techniques that have been discussed so far. These will now be explored.

## (b) Heuristics for Pruning the Combinatorial Space

The way the candidate generation problem is usually posed much larger ranges for carbon and hydrogen are generally admitted than are required for other elements. These large ranges for carbon and hydrogen greatly increase the number of combinations that need to be examined. It is therefore highly desirable to reduce the influence of the carbons and the hydrogens on the computation. Fortunately a simple and yet explicit technique can be used to factor out the combinatorial influence of these two elements on the computation.

To illustrate how this can be done let us for a moment consider how the combination $C_8H_{12}O_4Br_2N_2$ could be generated.

Let us further assume that we have a mechanism (see previous section) that has generated the partial combination $O_4Br_2N_2$. For this particular combination of oxygen, bromine and nitrogen it is necessary to examine all appropriate carbon and hydrogen combinations (e.g. $C_1H_2O_4Br_2N_2$, $C_2H_2O_4Br_2N_2$, ..., $C_{24}H_{50}O_4Br_2N_2$). Generation of this large set of combinations can be avoided by making a single simple computation based on the mass defect of the mass that is being fitted and the mass defect of the particular {O,N,Br} combination.

It is given that:

$$M_p = O_4Br_2N_2$$

and the accurate mass that is being fitted is

$$M_T = \text{accurate mass}$$

Now because carbon has an atomic weight of exactly 12.0000 amu we know that it makes *no contribution* to the mass defect of the accurate mass. If follows that the defect difference

$$D_D = D_T - D_p$$

must be due solely to contributions from the hydrogen defect $D_H$. Therefore the number of hydrogens $N_H$ in the composition is given by

$$N_H = D_D/D_H \qquad \text{(rounded to the nearest integer)}$$

The number of carbons $N_C$ is then given by

$$N_C = [M_T - (M_P + N_H + N_H \times D_H)]/12$$

With these modifications the size of the combinatorial space is markedly reduced. In the algorithm of Robertson and Hamming (1) the hydrogens are factored out but the carbons are left to contribute to the size of the combinatorial space.

There is yet another factoring technique that can generally be applied

to further reduce the dimensions of the combinatorial space.

It is based on the observation that the composition $CH_2$ (14.0156) and atomic weight of nitrogen (14.0031) are very close in their actual masses. We can therefore save on repeated calculation of the hydrogen and carbon contributions to the total composition by using the following procedure (assuming nitrogen is the last element added into the composition).

Before adding *any* nitrogens into the composition we calculate the "hydrogen contribution" $N_H'$.

If $C_8H_{12}O_4Br_2N_2$ is the final composition sought then at the stage when $O_4Br_2$ is generated it is usually necessary to examine all possible nitrogen contributions (e.g. $O_4Br_2N_1$, $O_4Br_2N_2$, $O_4Br_2N_3$, ...). Instead we can proceed as follows:

Let $M_P = O_4Br_2$

$M_T$ = high resolution mass data given

$M_D = M_T - M_P$

$D_D = D_T - D_P$

$N_H' = D_D/D_H$

$D_{dif} = M_C + 2 \times M_H - M_N$ ($CH_2$ and nitrogen defect difference

$N_C = (M_T - N_H' \times M_H)/12$

$N_H = (M_D - 12 \times N_C)$

$M_{T'} = M_P + N_H \times M_H + N_C \times M_C$

$D_{TP} = D_{T'} - D_P$

$N_N = D_{TP}/D_{dif}$

This procedure can be applied directly for up to seven nitrogens. Should a larger range for nitrogens be required it must be modified slightly to include an additional computation to decide whether there were $k$ nitrogens present or $(k+7)$ nitrogens present.

As it will be seen from the timing results the simpler stack mechanism together with the factoring out of the nitrogens, carbons,and hydrogens leads to a very efficient procedure.

## PARALLEL COMPUTATION OF ELEMENTAL COMPOSITIONS

In many applications the primary objective is to determine the elemental compositions for all masses in a complete high resolution spectrum. In this situation all compositions for smaller masses than the molecular weight are encompassed by the combinatorial space defined by the molecular weight. This situation can be exploited in the following way. Instead of making one pass through the combinatorial space for *each* mass it is possible to make just *one pass* through the combinatorial space for the complete set of masses. To do this a specially arranged defect table must be employed. The table is set up in such a way that the defect of the balance part (due only to carbon and hydrogen) when used as an index to the table will give a list of mass peaks that can be reached only by an integral number of hydrogens (all defects are multiplied by 1000 so that they can index the table which is also of size 1000). The problem with this method is that when the error in the system is expected to be of the order of ±0.003 amu its proximity to the hydrogen defect (0.0078 amu) makes the list of mass peaks associated with each reference to the table rather high. Consequently there is little gain in efficiency from the parallelism built into the computation. Tests showed however that when the expected error could be assumed to be of the order of 0.001 amu then the lists associated with each table reference are short and consequently there are significant gains to be had by doing the computation in parallel.

The stringent demands on accuracy of the strictly parallel method outlined above suggested that it may be better to opt for a pseudo-parallel implementation. The latter approach was implemented in the following way. Partial compositions were generated for all elements except hydrogen, carbon and nitrogen in the manner described for the single mass case. Then each mass peak was examined and the balance calculation with carbon, hydrogen and nitrogen was performed in the manner described earlier. The results for this method are given below in Table IV.

## RESULTS AND DISCUSSION

To evaluate the present proposals for speeding up elemental composition calculations an extensive series of tests and comparisons were made. The results of tests are summarized in tables I-IV.

The first method tested out was that of Robertson and Hamming (1). For this test Robertson's FORTRAN program,that is available from the Quantum Chemistry Program Exchange,was used without modification. Timing results for two different data sets run with this method are shown in table I. The data sets were deliberately chosen over different ranges to illustrate specific trends. The characteristics of the two data sets are summarized in table V.

Results for 7 and 8 elements for this method were not calculated because of the excessive CPU time required. Two observations can be made from the results in table I. The first is that with the increasing mass range the cost of the computation grows very rapidly. Secondly, with an increasing number of elements included in the calculation the cost also grows very rapidly. For more than six elements and a spectrum with an extended range the computation time becomes prohibitive.

Robertson and Hamming (1) point out that their algorithm is close to an order of magnitude faster than earlier general algorithms for elemental composition generation. Initial tests confirmed this observation. These algorithms were therefore not pursued further.

The results in tables II, III and IV summarize the various phases of the current alternative proposals. The simplest of these proposals (Table II) uses a stack mechanism in conjunction with a branch and bound algorithm. For this algorithm the carbons together with the hydrogens have been factored out of the calculation in the way described earlier. It can be seen that the effects of an increasing mass range and an increasing number of elements considered are far less drastic than for Robertson and Hamming's algorithm (1).

The results in table III show the additional gains in efficiency that can be made by factoring out the nitrogen as well as the carbon and the hydrogen.

In comparing the results in table I with those in table III it can be seen that the gains for the latter method are greatest when the mass range is large and when the number of elements included is high.

The results for the pseudo-parallel implementation of the algorithm are summarized in table IV. What is apparent from these results is that in general the gains in efficiency for this method are only small in a relative sense. This is because of the relatively large amount of time spent in the final calculation, which must still be carried out for each elemental combination,for all masses.

Copies of the program which is written in ANSI FORTRAN may be obtained from the authors. All programming tests were run on a UNIVAC 11Ø6 computer.

## CONCLUSIONS

An elemental composition generation system based on a branch and bound algorithm and some new pruning methods has been shown to provide large gains in efficiency over existing systems. The very short processing times that are possible with this new algorithm are important for processing high resolution mass spectra in real time applications.

## ACKNOWLEDGEMENTS

The authors wish to thank Mr M. Bruce and Dr J.K. MacLeod of the Australian National University for bringing to their attention Robertson and Hamming's algorithm. The present algorithm was developed before the authors became aware of this work. Mr C. MacDonald and Dr C. Whittle of CSIRO Canberra are also thanked for helpful discussions. The authors would also like to thank Miss Ann Titus for preparing the manuscript.

## LITERATURE CITED

1. A.L. Robertson, M.C. Hamming, Biomed. Mass Spectometry, 4, 203 (1977).

2. A.L. Burlingame, "Advances in Mass Spectrometry", (Editor E. Kendrick), Vol.4, 15, Institute of Petroleum, London (1968).

3. S.R. Shrader, "Introductory Mass Spectrometry", Allyn and Bacon, New York (1971).

4. J. Lederberg, "Computation of Molecular Formulas for Mass Spectrometry", Holden-Day, San Francisco (1964).

5. J.R. Chapman, "Computers in Mass Spectrometry", Academic Press, London (1978).

6. E.M. Reingold, J. Nievergelt, N. Deo, "Combinatorial Algorithms", Prentice-Hall, New Jersey (1977).

Table V

Mass range and peak count information for data sets examined.

| Spectrum Number | Number of Masses | Mass Range amu | Defect Range |
|---|---|---|---|
| 1 | 55 | 70 → 295 | 0.055 → 0.177 |
| 2 | 45 | 36 → 774 | -0.979 → 0.715 |

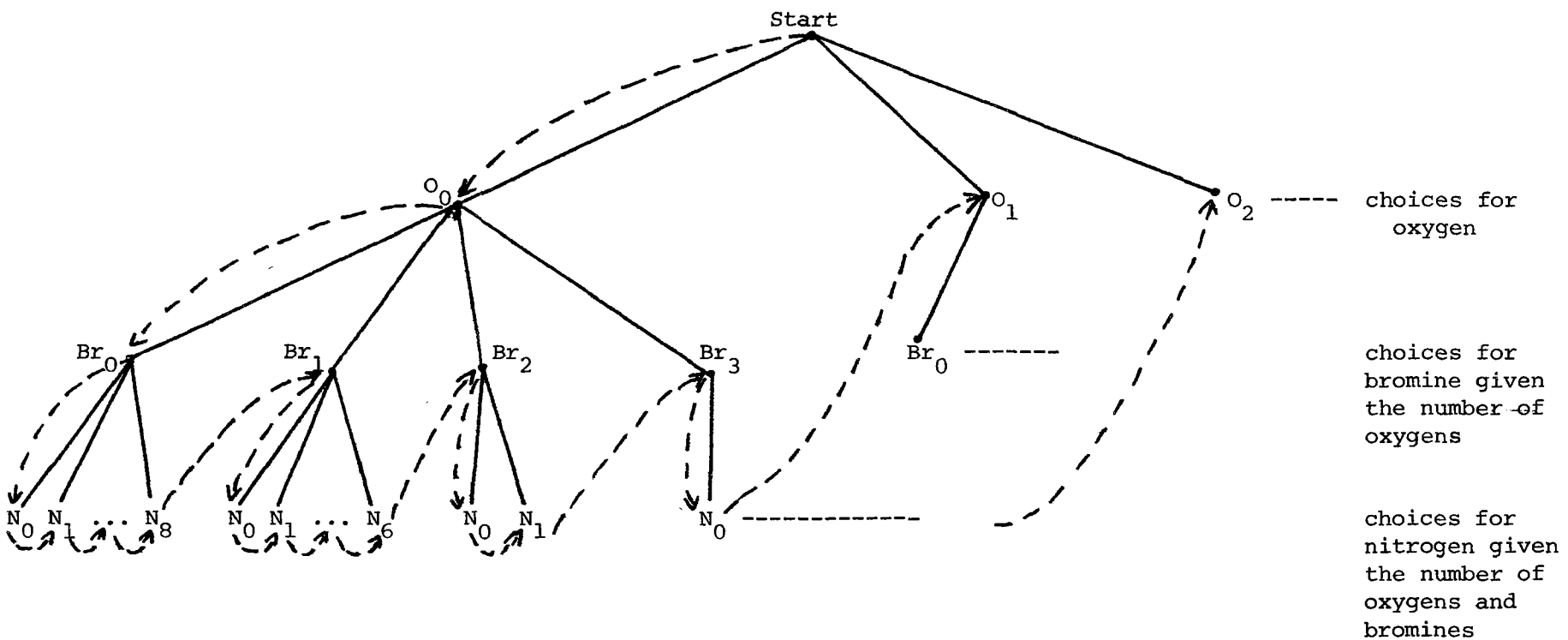
FIGURE CAPTIONS:

Figure 1: Example of mass stack operation

Figure 2: Illustration of tree-like growth of combinatorial space

Figure 3: Flowchart showing operation of stack mechanism.

| Root Composition | Mass Stack | Balance Deficit |
|---|---|---|
| $O_0Br_0N_0$ | 0 0 0 | 160 |
| $O_0Br_0N_1$ | 0 0 14 | 146 |
| $O_0Br_0N_2$ | 0 0 28 | 132 |
| ⋮ | ⋮ | ⋮ |
| $O_0Br_0N_8$ | 0 0 112 | 48 |
| $O_0Br_1N_0$ | 0 79 79 | 81 |
| $O_0Br_1N_1$ | 0 79 93 | 67 |
| ⋮ | ⋮ | ⋮ |
| $O_0Br_1N_5$ | 0 79 149 | 11 |
| $O_0Br_1N_6$ | 0 79 163 | overflow |
| $O_0Br_2N_0$ | 0 158 158 | 2 |
| $O_0Br_2N_1$ | 0 158 172 | overflow |
| $O_0Br_3N_0$ | 0 237 237 | overflow |
| $O_1Br_0N_0$ | 16 16 16 | 144 |
| $O_1Br_0N_1$ | 16 16 30 | 130 |
| ⋮ | ⋮ | ⋮ |


Start
$O_0$
$O_1$
$O_2$
$Br_0$
$Br_1$
$Br_2$
$Br_3$
$Br_0$
$N_0$ $N_1$ ... $N_8$
$N_0$ $N_1$ ... $N_6$
$N_0$ $N_1$
$N_0$
choices for oxygen
choices for bromine given the number of oxygens
choices for nitrogen given the number of oxygens and bromines

Parameters:

NE: Array giving current number of each element

ST: Mass stack array

EP: Pointer to current element

ME: Array with maximum permissable number of each element

W: Atomic mass of each element

M: Current desired mass peak

LE: Number of last element

P: Auxiliary element pointer

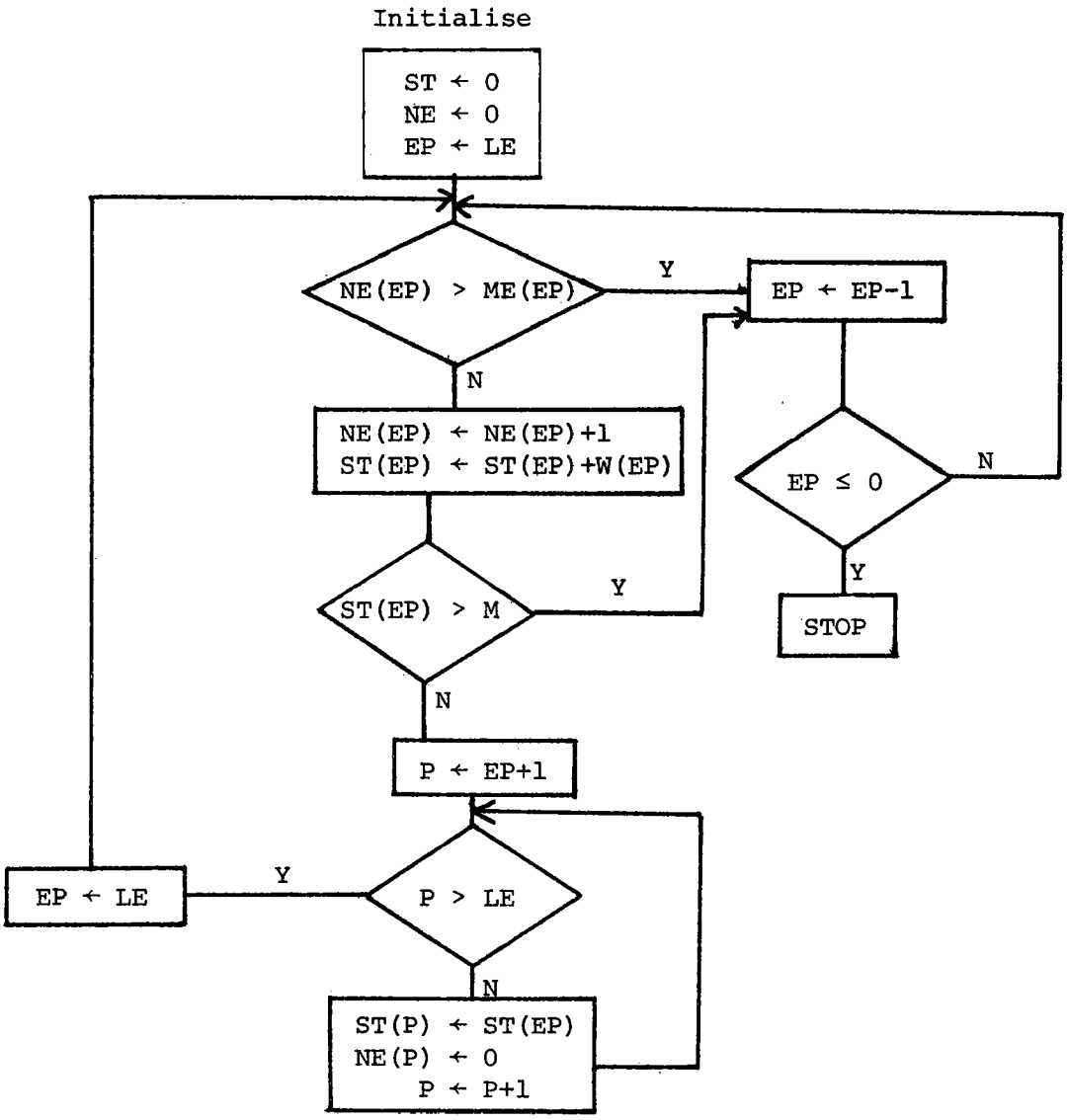


Table I. Timing results for Robertson and Hamming's Q.C.P.E. Program.

| No. elements | Spectrum 1 C.P.U. Time(s) | Formulae Tested | Spectrum 2 C.P.U. Time(s) | Formulae Tested |
|---|---|---|---|---|
| 4 | 2.42 | 2679 | 22.00 | 26153 |
| 5 | 7.18 | 6019 | 145.89 | 147875 |
| 6 | 16.40 | 12003 | 899.04 | - |

Table II. Timing results for Simple Stack Mechanism, with Carbons and Hydrogens factored out.

| No. elements | Spectrum 1 C.P.U. Time(s) | Spectrum 2 C.P.U. Time(s) |
|---|---|---|
| 4 | 0.32 | 0.24 |
| 5 | 0.88 | 0.89 |
| 6 | 2.12 | 3.31 |
| 7 | 2.86 | 8.69 |
| 8 | 8.49 | 37.54 |

Table III. Timing results for Stack Mechanism with C, H, N in one step.

| No. elements | Spectrum 1 C.P.U. Time(s) | Formulae Tested | Spectrum 2 C.P.U. Time(s) | Formulae Tested |
|---|---|---|---|---|
| 4 | .08 | 495 | .05 | 405 |
| 5 | .24 | 2241 | .19 | 1845 |
| 6 | .65 | 6786 | .64 | 7344 |
| 7 | .91 | 10053 | 1.78 | 21591 |
| 8 | 2.81 | 32580 | 7.48 | 94455 |

Table IV. Timing results for Pseudo - parallel implementation.

| No. elements | Spectrum 1 C.P.U. Time(s) | Spectrum 2 C.P.U. Time(s) |
|---|---|---|
| 4 | .07 | .05 |
| 5 | .22 | .18 |
| 6 | .53 | .56 |
| 7 | .72 | 1.57 |
| 8 | 2.20 | 6.55 |