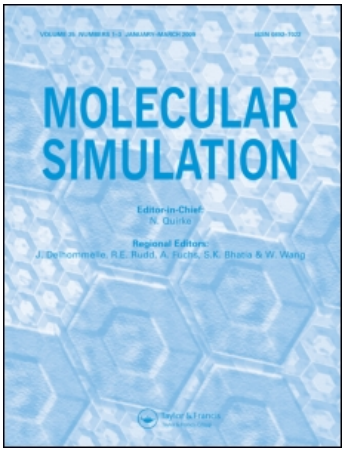

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Boolean Scheme for Programming Trial Moves That Involve Molecule Insertion and Removal in Monte Carlo Simulation

Javier Carrero-Mantilla[a]; Mario Llano-Restrepo[a]
[a] School of Chemical Engineering, Universidad del Valle, Cali, Colombia

Online publication date: 26 October 2010

To cite this Article Carrero-Mantilla, Javier and Llano-Restrepo, Mario(2003) 'Boolean Scheme for Programming Trial Moves That Involve Molecule Insertion and Removal in Monte Carlo Simulation', Molecular Simulation, 29: 1, 23 — 28
To link to this Article: DOI: 10.1080/0892702031000065700
URL: http://dx.doi.org/10.1080/0892702031000065700

Molecular Simulation, 2003 Vol. 29 (1), pp. 23–28

Boolean Scheme for Programming Trial Moves that Involve Molecule Insertion and Removal in Monte Carlo Simulation

JAVIER CARRERO-MANTILLA and MARIO LLANO-RESTREPO*

School of Chemical Engineering, Universidad del Valle, Apartado 25360, Cali, Colombia

(Received April 2002; In final form June 2002)

A Boolean scheme is presented for programming trial moves that involve molecule insertion and removal in both grand canonical (GCMC) and Gibbs ensemble (GEMC) Monte Carlo simulation methods. The Boolean scheme makes use of logical data type arrays for keeping track of molecules being inserted and removed, instead of the integer index array technique of the traditional Nicholson's scheme. An exploratory evaluation of the relative computational performance of the two schemes is presented. For GCMC simulation of simple fluids, Nicholson's scheme appears to be more efficient than the Boolean scheme. In contrast, for GEMC simulation of vapor–liquid equilibria (VLE) in binary systems, the Boolean scheme seems to become more efficient as the complexity of the intermolecular potential increases. However, considering the limitations intrinsic to any evaluation of computational performance, the two schemes should be regarded as alternative tools to be tried for the particular application of interest to the user.

Keywords: Simulation methodology; Monte Carlo simulation; Grand canonical ensemble; Gibbs ensemble; Trial moves; Insertion and removal

INTRODUCTION

Both grand canonical (GCMC) [1–4] and Gibbs ensemble (GEMC) [5–14] Monte Carlo simulation methods involve trial moves based on molecule insertion and removal. For instance, the transfer trial move of a molecule of a given species from box I to box II in the GEMC method involves the removal of a molecule of that species in box I and the insertion of a molecule of the same species in box II.

The integer index array technique originally devised by Nicholson [1] for the GCMC method, and explained in detail in the classic textbook by Allen and Tildesley [2], may be easily and successfully extended to the GEMC method for programming that kind of trial moves, as shown elsewhere.

Even though an insightful presentation of both GCMC and GEMC methods is found in the textbook by Frenkel and Smit [3,13], no specific array techniques were discussed by those authors to handle the trial moves involving molecule insertion and removal. In the recent textbook by Sadus [4], no algorithm is specified for handling the molecule insertion or removal moves though it is stated that the procedure is bound tightly to the programming language used, in such a way that in FORTRAN the scheme should invariably involve manipulating array indices.

Even though Nicholson's technique works perfectly well and is probably used by many authors, it is possible to devise an alternative scheme involving an array with logical data type elements (whose values shift from `.TRUE.` to `.FALSE.`) in order to keep track of those molecules inserted or removed in a given box during the course of a simulation run. In this work, such a Boolean scheme is proposed and compared on an exploratory basis with Nicholson's scheme.

NICHOLSON'S SCHEME FOR THE GCMC METHOD

In the Nicholson's technique [1,2] for the GCMC method, an array (named `LOCATE`) with integer data type elements, is used in order to keep track of those molecules that are currently active (or "alive") in the simulation box. The size of the array is equal to

*Corresponding author. Tel.: +57-2-3312935. Fax: +57-2-3392335. E-mail: mllano@mafalda.univalle.edu.co

ISSN 0892-7022 print/ISSN 1029-0435 online © 2003 Taylor & Francis Ltd
DOI: 10.1080/0892702031000065700

the maximum number of molecules expected in the box. At the beginning of the simulation run, all the elements of the array are set to zero and then initialised by running a loop (with upper index equal to the initial number of molecules in the box) in which the running integer index is assigned to the element of the array located at a position equal to the value of the index. At any stage of the simulation run, the array consists of two areas: the "alive" area that contains the indices of the molecules currently active, and the "dead" area that contains not only the indices of the molecules currently inactive in that box but also as many zero entries as required to fill up the declared size of the array.

When the attempted removal of a randomly chosen molecule is found to be successful, according to the required acceptance criterion, the array is updated by closing it up (i.e. by moving one position backwards those indices in the active area of the array that are located to the right of the index of the molecule being removed), by moving the index of the removed molecule to the position in the array corresponding to the current number of alive molecules (which is by then the first position in the "dead" area of the array), and by decreasing by one the number of active molecules.

When the attempted insertion of a molecule is found to be successful, the array is updated by increasing the number of active molecules by one and by giving the inserted molecule an index equal to the first element of the "dead" area, if that element is different from zero. If the first element of the "dead" area is equal to zero, the inserted molecule is given an index equal to the updated number of active molecules.

At any stage of a GCMC simulation run, the configurational energy is recalculated by running a loop (with upper index equal to the number of molecules currently "alive" in the simulation box) in which the indices of the active molecules are retrieved for identifying the current positions of the molecules.

The code for updating and reordering the array `LOCATE` for Monte Carlo simulation in the GCMC, using Nicholson's technique, is listed as program F.14 in Appendix F of the book by Allen and Tildesley [2].

NICHOLSON'S SCHEME FOR THE GEMC METHOD

Since Nicholson's technique was specifically intended for use with the GCMC method [1], at a time when the GEMC method had not yet been proposed, an adaptation has to be made in order to implement that technique for GEMC simulation of vapor–liquid equilibria (VLE) of a binary system. For that purpose, a total of four arrays of the type `LOCATE` are required, one for each of the two components (A and B) in each of the two simulation boxes (I and II).

When the attempted transfer of a randomly chosen molecule of component A (usually the species of smaller molecules) from box I (the randomly chosen supplying box) to box II (the receiving box) is found to be successful, according to the required acceptance criterion, the index of that molecule, which is located in the array named `LOCATE_A_I`, is copied to the position next to the last entry of the active area of the array named `LOCATE_A_II`. The array `LOCATE_A_I` is updated by closing it up (that is, by moving one position backwards those indices in the active area of the array that are located to the right of the index of the molecule being removed), by switching to zero the element of the array located at the position corresponding to the current number of alive molecules, and by decreasing by one the number of active molecules in box I. The array `LOCATE_A_II` is updated by increasing by one its number of active molecules. Since the index of a removed molecule is transferred from one array to another, the "dead" areas of the four arrays are always filled with zero entries.

In a modified version of the GEMC simulation method [8,9], identity exchange trial moves are performed in order to make possible the insertion of a molecule of the larger species in the denser phase by using the place already occupied by a molecule of the smaller species. This kind of trial move can be regarded as two simultaneous transfer trial moves (one move for the molecule of the larger species, and the other move for the molecule of the smaller species), and can be easily coded by following the steps explained above for updating each of the four arrays of type `LOCATE`.

At any stage of a GEMC simulation run, the configurational energy is recalculated in each box by running a loop (with upper index equal to the number of molecules currently "alive" in the given box) in which the indices of the active molecules are retrieved for identifying the current positions of the molecules.

THE BOOLEAN SCHEME FOR THE GCMC METHOD

The Boolean scheme makes use of an array with logical data type elements instead of the integer indices used in Nicholson's scheme. The logical values `.TRUE.` or `.FALSE.` in the array `ALIVE` indicate the presence or absence, respectively, of a given molecule in the simulation box. That is, if element k in the array `ALIVE` is equal to the logical value `.TRUE.` that means that molecule k is currently

active in the simulation; if element *k* is equal to the logical value .FALSE. that means that molecule *k* is currently inactive in the simulation. As in Nicholson's scheme, the size of the array is equal to the maximum possible number of molecules in the simulation.

A trial move for removing a molecule can be easily coded using the Boolean scheme. Let us suppose that among all of the "alive" molecules currently present in the simulation box, the *k*-th "alive" molecule has randomly been chosen for the removal attempt, by generating a random number in the interval [0,1] and multiplying that number by the total number of currently active molecules in the box. In order to program the removal trial move, it is necessary to locate the position of the chosen active molecule in the array ALIVE (which contains the two kinds of logical values). That position is located by calling the following function (coded in the standard Fortran 95 programming language):

```
INTEGER FUNCTION TruePos (ArrayIn,
 req_pos)
  IMPLICIT NONE
  LOGICAL, DIMENSION (:), INTENT
    (IN) :: ArrayIn
  INTEGER, INTENT (IN) :: req_pos
  INTEGER :: indx, sum_t
  LOGICAL :: counting
  sum_t = 0; counting = .TRUE.;
   indx = 0
  DO WHILE (counting)

  indx = indx + 1
  IF (ArrayIn (indx) = = .TRUE.)
    sum_t = sum_t + 1
  IF (sum_t = = req_pos)
    counting = .FALSE.

  END DO
  TruePos = indx
  END FUNCTION
```

The statement for calling the function would be written as

```
index = TruePos (ALIVE, k)
```

where the integer variable index stores the value of the inquired position.

Once that position has been found, the following logical (Boolean) assignment statement is required to remove the *k*-th "alive" molecule in the simulation box:

```
ALIVE (index) = .FALSE.
```

A trial move for inserting a molecule can also be easily coded using the Boolean scheme. For that purpose, the first position in the array ALIVE with the logical value .FALSE. is found by calling the following function (coded in the standard Fortran 95 programming language):

```
INTEGER FUNCTION FirstFalse (ArrayIn)
  IMPLICIT NONE
  LOGICAL, DIMENSION (:), INTENT
    (IN) :: ArrayIn
  INTEGER :: indx
  LOGICAL :: counting
  counting = .TRUE.; indx = 0
  DO WHILE (counting)

  indx = indx + 1
  IF (ArrayIn (indx) = = .FALSE.)
   THEN
  counting = .FALSE.
  END IF

  END DO
  FirstFalse = indx
  END FUNCTION
```

The statement for calling the function would be written as

```
index = FirstFalse (ALIVE)
```

where the integer variable index stores the value of the inquired position.

Once that position has been found, the following logical (Boolean) assignment statement is required to insert the molecule in the simulation box:

```
ALIVE (index) = .TRUE.
```

At any stage of a GCMC simulation run, the configurational energy is recalculated by running a loop (with upper index equal to the declared size of the array ALIVE) in which prior to retrieving the current positions of the molecules, the active molecules are successively identified by inquiring whether the element in the array corresponding to the position given by the value of the running index, is equal to the logical value .TRUE.

THE BOOLEAN SCHEME FOR THE GEMC METHOD

As in Nicholson's scheme, an array is required for each species in each of the simulation boxes. In the Boolean scheme, the elements of the array (named BELONGTO) are the logical value .TRUE. or .FALSE. instead of the integer indices used in Nicholson's technique. The size of the array is equal to the maximum possible number of molecules in the simulation. For instance, for the GEMC simulation of the VLE of a binary system, with an initial number of 256 molecules in each box, a total of four arrays of the type BELONGTO (with 512 elements each) are

required, one for each component in each box. This is the same requirement as for Nicholson's scheme.

If element *k* in the array `BELONGTO_A_I` (defined for component A in box I) is equal to the logical value `.TRUE.` that means that molecule *k* of component A belongs to box I. At the same time, in the array `BELONGTO_A_II` (defined for component A in box II), element *k* is equal to the logical value `.FALSE.` and that means that molecule *k* of component A does not belong to box II.

The transfer trial move of a molecule of component A (usually the species of smaller molecules) from box I (the randomly chosen supplying box) to box II (the receiving box), can be easily coded using the Boolean scheme. Let us suppose that among all of the "alive" molecules currently present in box I, the *k*-th "alive" molecule has randomly been chosen for the transfer attempt, by generating a random number in the interval [0,1] and multiplying that number by the total number of currently active molecules in that box. In order to program the transfer trial move, it is necessary to locate the position of the chosen active molecule in the array `BELONGTO_A_I` (which contains the two kinds of logical values). That position is located by calling the TruePos function as follows:

```
index = TruePos (BELONGTO_A_I, k)
```

where the integer variable `index` stores the value of the inquired position.

Once that position has been found, the following two logical (Boolean) assignment statements are required to make the transfer trial move of the *k*-th "alive" molecule from box I to box II:

```
BELONGTO_A_I (index) = .FALSE.
BELONGTO_A_II (index) = .TRUE.
```

In a modified version of the GEMC simulation method [8,9], identity exchange trial moves are performed in order to make possible the insertion of a molecule of the larger species in the denser phase by using the place already occupied by a molecule of the smaller species. Let us suppose that a molecule of component A in box I, with the position in the array `BELONGTO_A_I` indicated by the value of the variable `index_A`, exchanges identity with a molecule of component B in box II, with the position in the array `BELONGTO_B_II` indicated by the value of the variable `index_B`.

The following four logical assignment statements are required to make the identity exchange trial move:

```
BELONGTO_A_I (index_A) = .FALSE.
BELONGTO_A_II (index_A) = .TRUE.
BELONGTO_B_I (index_B) = .TRUE.
BELONGTO_B_II (index_B) = .FALSE.
```

At any stage of a GEMC simulation run, the configurational energy is recalculated in each box by running a loop (with upper index equal to the declared size of the array `BELONGTO`) in which prior to retrieving the current positions of the molecules, the active molecules are successively identified by inquiring whether the element in the array corresponding to the position given by the value of the running index, is equal to the logical value `.TRUE.`

EXPLORATORY EVALUATION OF COMPUTATIONAL PERFORMANCE

For evaluating the relative computational performance of the two schemes, two codes were written for each study case considered in this work. In one of the codes the Boolean scheme was implemented, and the other code made use of Nicholson's scheme. The two codes were exactly the same regarding all statements different from those required for insertion and removal trial moves (in the case of GCMC simulations) and transfer and identity exchange trial moves (in the case of GEMC simulations). The two codes were written in the standard Fortran 95 programming language, compiled using version 1.60 of the Salford Win32 F95 compiler, and executed in a PC with a Pentium IV/1.5 GHz processor.

For a given simulation run, the CPU time required for each code was recorded using the compiler built-in CPU_TIME function. Let t_B be the CPU time required for the code with the Boolean scheme and let t_N be the CPU time required for the code with Nicholson's scheme. The CPU time percentage difference $\Delta t_{B/N}$ for the code with the Boolean scheme relative to the code with Nicholson's scheme is defined as:

$$\Delta t_{B/N} = 100\left(\frac{t_B - t_N}{t_N}\right)$$

The Lennard–Jones (LJ) fluid was chosen for the test runs in the GCMC. In Table I, results for $\Delta t_{B/N}$ are shown at a reduced temperature $T^* = kT/\epsilon = 1.5$. The resulting values of the average reduced number density $\rho^* = \langle N\rangle/V^*$ (where $V^* = V/\sigma^3$ is the reduced volume of the simulation box, with ϵ and

TABLE I CPU time percentage differences for the Boolean scheme relative to Nicholson's scheme for GCMC simulation with the Lennard–Jones fluid

T^*	$\langle N\rangle/V^*$	$\Delta t_{B/N}$ (%)
1.5	0.417	+0.8
1.5	0.510	+3.8
1.5	0.600	+3.0
1.5	0.703	+3.4
1.5	0.806	+3.3
1.5	0.904	+2.9

TABLE II CPU time percentage differences for the Boolean scheme relative to Nicholson's scheme for GEMC simulation of VLE of the system argon/methane

T (K)	P (bar)	$\Delta t_{B/N}$ (%)
140	7.00	+9.7
140	16.00	+12.4
140	24.00	+12.2
178	32.10	+11.7
178	39.85	+15.7
178	49.76	+22.8

σ as the LJ energy and size parameters), are also listed in Table I. For this choice of intermolecular potential, the Boolean scheme required on the average 2.9% more computer time than Nicholson's scheme. Therefore, Nicholson's scheme appears to be more efficient than the Boolean scheme for GCMC simulation of simple fluids.

The binary systems argon/methane and methane/ethane were chosen for the test runs in the GEMC. GEMC simulations of VLE were carried out, modelling argon and methane as one-centre LJ fluids and ethane as a two-centre LJ fluid, with the energy and size parameters ϵ and σ given by Panagiotopoulos for argon [8] and Vrabec and Fischer for methane [15,16] and ethane [16]. GEMC simulation runs were initiated with 256 molecules in each box, and consisted of 2000 cycles for equilibration and 3000 cycles for production. Every cycle included as many translational (and orientational, for ethane molecules) trial moves as molecules were contained in each box, followed by one volume change trial move, 250 argon or methane transfer trial moves and 50 methane to ethane identity exchange trial moves. These specifications are typical of GEMC simulation studies already published in the literature [17].

As shown in Table II, for six sets of VLE conditions of the system argon/methane, the Boolean scheme required on the average 14.1% more computer time than Nicholson's scheme. In contrast, as shown in Table III, for 11 sets of VLE conditions of the system methane/ethane, the Boolean scheme appears to be on the average 18.8% more efficient than Nicholson's scheme. These results seem to indicate that for GEMC simulation of VLE in binary systems, the proposed Boolean scheme tends to become more efficient than the traditional Nicholson's scheme as the complexity of the intermolecular potential increases.

TABLE III CPU time percentage differences for the Boolean scheme relative to Nicholson's scheme for GEMC simulation of VLE of the system methane/ethane

T (K)	P (bar)	$\Delta t_{B/N}$ (%)
172.0	2.21	−21.2
172.0	12.41	−17.7
172.0	20.68	−11.5
213.7	27.58	−15.5
220.0	40.00	−18.1
220.0	50.00	−19.4
240.0	10.00	−19.5
240.0	20.00	−15.0
250.0	15.63	−27.8
250.0	40.83	−19.1
250.0	66.57	−21.9

CONCLUSION

Because of the limited number of intermolecular potentials of interaction that have been considered for this exploratory evaluation of computational performance and due to the limitations intrinsic to any evaluation of this kind, such as the actual dependence of the CPU time differences on items like computer architecture, processor speed, programming language, compiler version and code structure, it is not possible from this work to draw a clear-cut conclusion on the relative superiority of any of the two schemes considered. For this reason, the two schemes should be regarded as alternative tools to be explored in more detail and tried for the particular application of interest by the concerned user of GCMC and GEMC simulation methods.

Acknowledgements

Financial support from the School of Chemical Engineering, Universidad del Valle, Colombia, for this work (part of a larger GEMC simulation project), is gratefully acknowledged by the authors.

References

[1] Nicholson, D. (1984) "Grand ensemble Monte Carlo", *CCP5 Quarterly* **11**, 19.
[2] Allen, M.P. and Tildesley, D.J. (1987) *Computer Simulation of Liquids* (Clarendon Press, Oxford), pp. 129–30, pp. 343–344.
[3] Frenkel, D. and Smit, B. (1996) *Understanding Molecular Simulation. From Algorithms to Applications* (Academic Press, San Diego, USA), pp. 113–120, pp. 395–398.
[4] Sadus, R.J. (1999) *Molecular Simulation of Fluids. Theory, Algorithms and Object-Orientation* (Elsevier, Amsterdam), pp. 293–296.
[5] Panagiotopoulos, A.Z. (1987) "Direct determination of phase coexistence properties of fluids by Monte Carlo simulation in a new ensemble", *Molecular Physics* **61**, 813.
[6] Panagiotopoulos, A.Z., Quirke, N., Stapleton, M. and Tildesley, D.J. (1988) "Phase equilibria by simulation in the Gibbs ensemble. Alternative derivation, generalization and application to mixture and membrane equlibria", *Molecular Physics* **63**, 527.
[7] Panagiotopoulos, A.Z. and Stapleton, M.R. (1989) "The Gibbs method for molecular-based computer simulations of phase equilibria", *Fluid Phase Equilibria* **53**, 133.
[8] Panagiotopoulos, A.Z. (1989) "Exact calculations of fluid phase equilibria by Monte Carlo simulation in a new statistical ensemble", *International Journal of Thermophysics* **10**, 447.
[9] De Pablo, J.J. and Prausnitz, J.M. (1989) "Phase equilibria for fluid mixtures from Monte Carlo simulation", *Fluid Phase Equilibria* **53**, 177.
[10] Smit, B., de Smedt, Ph. and Frenkel, D. (1989) "Computer simulations in the Gibbs ensemble", *Molecular Physics* **68**, 931.

[11] Smit, B. and Frenkel, D. (1989) "Calculation of the chemical potential in the Gibbs ensemble", *Molecular Physics* **68**, 951.
[12] Panagiotopoulos, A.Z. (1992) "Direct determination of fluid phase equilibria by simulation in the Gibbs ensemble: a review", *Molecular Simulation* **9**, 1.
[13] Frenkel, D. and Smit, B. (1996) *Understanding Molecular Simulation. From Algorithms to Applications* (Academic Press, San Diego, USA), pp. 183–204, pp. 399–407.
[14] Sadus, R.J. (1999) *Molecular Simulation of Fluids. Theory, Algorithms and Object-Orientation* (Elsevier, Amsterdam), pp. 339–348.
[15] Vrabec, J. and Fischer, J. (1995) "Vapour-liquid equilibria of mixtures from the *NPT* plus test particle method", *Molecular Physics* **85**, 781.
[16] Vrabec, J. and Fischer, J. (1996) "Vapor–liquid equilibria of binary mixtures containing methane, ethane, and carbon dioxide from molecular simulation", *International Journal of Thermophysics* **17**, 889.
[17] Liu, A. and Beck, T.L. (1998) "Vapor–liquid equilibria of binary and ternary mixtures containing methane, ethane and carbon dioxide from Gibbs ensemble simulation", *Journal of Physical Chemistry B* **102**, 7627.