J Math Chem (2014) 52:627–645
DOI 10.1007/s10910-013-0283-z

ORIGINAL PAPER

Self avoiding walk trees and laces

A. S. Padmanabhan · Susamma Jacob

Received: 1 July 2013 / Accepted: 29 October 2013 / Published online: 10 November 2013

Abstract This paper is divided into two sections. In the first section we describe the use depth first search trees and breadth first search trees in understanding self avoiding walks and restricted random walks. While we derive our results for Euclidean lattices the method is totally general and can be used for other lattices as well. In the second section derive an expression for the number of laces in the lace expansion of a memory four restricted random walk.

Keywords Depth first search trees · Self avoiding walks · Lace expansion

1 Introduction

The Statistical mechanics of polymer molecules in dilute solution was first explained by the Nobel Laureate Paul J. Flory [1]. He considered single polymer chains in isolation in (a) good solvent and (b) poor solvent. In good solvent the only interaction between the atoms of the polymer chain is repulsion and this is best modelled by the self avoiding walk. This explains in a satisfactory way the critical exponent of mean square end to end distance (radius of gyration) obtained from experiment. In poor solvent there are two types of interaction between atoms in the polymer chain,

A. S. Padmanabhan (✉) · S. Jacob
School of Chemical Sciences, Centre for High Performance Computing,
Mahatma Gandhi University, Priyadarshini Hills P.O.,
Kottayam 686560, Kerala, India
e-mail: as.padmanabhan@gmail.com

A. S. Padmanabhan · S. Jacob
Department of Mathematics, Bishop Moore College,
Mavelikara, Alapuzha, Kerala, India

repulsion and attraction. This is best modelled by the use empirical potentials such as Lennard Jones potential. The salient feature of the use of this potential in the calculation of the mean square end to end distance is the existence of a phenomenon called polymer collapse that is also observed experimentally [2,3].

Polymerization commonly occurs by means of two reaction mechanisms (a) addition (b) condensation. Calculations of the mean square end to end distance growing polymer chains undergoing addition polymerization in good solvent (repulsion interaction only) were extensively carried out about thirty years ago. However, no calculations have been carried out for growing polymer chains using potentials for both attraction and repulsion. Similarly no calculations have been carried out for polymer chains growing by the condensation mechanism, in both cases i.e. with (a) repulsion only (b) both attraction and repulsion. Nisha Satheesh recently carried out calculations for growing polymer chains (real) in poor solvent, i.e. with both attraction and repulsion [4,5]. Her findings were very interesting because she was able to observe the absence of polymer collapse in polymer chains growing by the addition mechanism. However, she observed the collapse phenomena in polymer chains growing by the condensation mechanism.

Self avoiding walks have been studied for many decades now [6]. It is generally assumed that a search for exact formula is beyond the reach of current techniques and the thrust has been on analysing asymptotic behaviour as the number of steps becomes very large. However, it has been conjectured that the generating function for the cardinality of the set of self-avoiding walks is non D finite [7,8]. It would therefore be more appropriate to treat a self avoiding walk of n steps as a restricted random walk of n steps with memory n and then study such walks as n increases [6,9,10].

In this paper we discuss two approaches to study the problem in the manner explained in the previous paragraph. Both methods have a strong connection with those used in the enumeration of graphs. In the first method we explore the use of Depth first search trees commonly used by computer scientists [11]. Computer programs using the Depth first search type algorithms commonly follow what is known as the back tracking paradigm. Minor modifications to the Breadth first type algorithm discussed here gives us parallel or distributed computing programs using the divide and conquer paradigm. This method proves to be very useful as we can conjecture results by tracing the route used by the algorithm in a computer and then use standard techniques to prove the conjecture. In the next section we note that the generating function $\chi^{\tau}(z)$ for the cardinality of the set of memory τ walks for $\tau = 0, 2,$ and 4 is given by:

$$\chi^0(z) = \frac{1}{1-(2d)\,z} \tag{1.1}$$

$$\chi^2(z) = \frac{1+z}{1-(2d-1)\,z} \tag{1.2}$$

$$\chi^4(z) = \frac{1+2z^2+((2d-1)\,z^3}{1-(2d-2)\,z-(2d-2)\,z^2-z^3} \text{ (Theorem 2.2)} \tag{1.3}$$

It is hoped that the methods used in Theorem 2.2 should enable us to derive generating functions for $\tau = 6, 8$, and 10 walks. This in turn should enable us to conjecture a generating function for $2n$ step walks with $\tau = 2n$.

Another possible application of this method is in obtaining a better understanding of μ, the connective constant for self avoiding walks [6]. It is well known that restricted random walks are Markovian whereas self avoiding walks are non Markovian. Restricted random walks of memory τ can be described by a transition matrix whose largest eigen value equals μ^{τ} [9,10,12]. This method has been used exactly estimate μ for memory 4 walks and to obtain upper estimates for walks with larger memory [12]. A very accurate estimate of the connective constant for three dimensional lattice has been recently published [13]. The use of search trees, however, would make it possible to obtain μ rigorously for larger values of memory. This is clear from a close examination of the method. For example in obtaining Theorem 2.2 and Lemma 2.7 we compare memory 2 trees with memory 4 trees and remove the $2d(2d-2)$ vertices (where d stands for the dimension of the lattice) and all their descendants at level 4 as a result of 4 step reversals. Similarly we remove $(2d-2)(2d)(2d-2)$ vertices and all their descendants at level 5. If this method has to be extended to memory 6 walks we would on comparison of memory 4 trees with memory 6 trees have to remove $2\,(2d)\,(2d-2) + (2d)\,(2d-2)\,(2d-3) + 3\,(2d)\,(2d-2)\,(2d-4)$ vertices and all their descendants at level 6 as a result of six step reversals. Similarly we remove $2\,(2d-2)\,(2d)\,(2d-2) + (2d-3)\,(2d)\,(2d-2)\,(2d-3) + 3\,(2d-2)\,(2d)\,(2d-2)\,(2d-4) - (2d-3)\,(2d)\,(2d-2)$ vertices and all their descendants at level 7 etc. (Here the fourth term is an inclusion exclusion term representing vertices and their descendants that have been removed by both memory 4 and memory 6 DFS trees). Computer programs for restricted random walks and self avoiding using symmetry described in earlier work can be used in the case of very high values of the number of steps [14,15].

Tree search methods are useful in two other contexts:

(1) Growing self avoiding walk models which were proposed to understand both growth processes in addition and condensation polymerization [16–22] can be studied by tree search methods [22] in a unified manner.
(2) The collapse of linear single polymer chains has been studied by assigning a factor $\lambda = e^{1/T}$ to each pair neighbouring sites on the lattice using a transfer matrix method [23,24]. This method is however useful only for two dimensional walks. The DFS tree method, however, can be used for any dimension and by assigning the factor $\lambda = e^{1/T}$ to each node in the tree corresponding to a pair of neighbouring sites on the lattice visited by the self avoiding walk. This will enable easy computation of collapse in any dimension for self avoiding walks with uniform distribution and also for growing self avoiding walks. Work is currently in progress in this area and it is hoped that this will facilitate rigorous analysis of the collapse phenomenon.

The second method discussed is the lace expansion method which has been in existence for many years now [6]. We are primarily interested in this method for the study of moments of $2n$ step walks with $\tau = 2n$.

We conclude this introduction by stating some standard definitions and notation (We follow Madras and Slade [6] for the theory of self avoiding walks and the lace expansion and Bondy and Murthy [11] for Graph theory).

An n step self avoiding walk on d dimensional hyper cubic lattice Z^{d}, beginning at the origin, is defined as a sequence of sites $(\omega(0), \omega(1), \ldots, \omega(n))$ with $\omega(0) = 0$ satisfying $|\omega(j+1) - \omega(j)| = 1$, *and* $\omega(i) \ddagger \omega(j)$ for all $i \ddagger j$ and where $|x|$ *refers to the Euclidean norm* $\sqrt{x.x}$. In the above definition we define $X^{(j)}$ to be the unit vector $\omega(j+1) - \omega(j)$ assuming $2d$ possible values minus the values that case self intersection [1].

C_n refers to the cardinality of the set of n step self avoiding walks.

A square refers to a four step self avoiding walk having the unit vectors $X^{(1)}$, $X^{(2)}$, $\mathrm{X}^{(3)}$, and $\mathrm{X}^{(4)}$ with $X^{(1)}$ orthogonal to $X^{(2)}$ and $X^{(3)} = -X^{(1)}$ and $X^{(4)} = -X^{(2)}$. Similarly a hook refers to a three step self avoiding walk with the second step (vector) orthogonal to the first and third steps(vectors) and the first and third steps (vectors) in opposite directions i.e., $X^{(1)}$, $X^{(2)}$, $\mathrm{X}^{(3)}$ with $X^{(1)}$ orthogonal to $X^{(2)}$ and $X^{(3)} = -X^{(1)}$.

The connective constant $\mu = \lim_{n\to\infty} C_n^{1/n}$ is known to exist [1].

The generating function $\chi(z) = \sum_{n=0}^{\infty} C_n z^n$ has radius of convergence

$$z_c = \left[\lim_{n\to\infty} C_n^{1/n}\right]^{-1} = \frac{1}{\mu}[1].$$

We define a memory τ walk as a walk which is self avoiding only over a finite time scale τ (or memory τ). In other words:

An n step memory τ walk on d dimensional hyper cubic lattice Z^{d}, beginning at the origin, is defined as a sequence of sites $(\omega(0), \omega(1), \ldots, \omega(n))$ with $\omega(0) = 0$ satisfying $|\omega(j+1) - \omega(j)| = 1$, *and* $\omega(i) \ddagger \omega(j)$ whenever $0 < |i - j| \leq \tau$ and where $|x|$ *refers to the Euclidean norm* $\sqrt{x.x}$. In this paper we will be concerned primarily with memory 0 (simple random walks), memory 2 (walks with exclusion of immediate reversals) and memory 4 walks.

$C_{n,\tau}$ will refer to the number of memory τ walks, $\mu^{\tau} = \lim_{n\to\infty} C_{n,\tau}^{1/n}$ and $\chi^{\tau}(z) = \sum_{n=0}^{\infty} C_{n,\tau} z^n$

In Theorem 2.1 and Lemma 2.7 and Theorem 2.2 of the next section we will without loss of generality use C_n to refer to $C_{n,4}$, mainly because the logic used in these Theorems and the Lemma will be valid for higher memory walks. This is proposed to be investigated in future.

Appending a step (unit vector) $\mathrm{X}^{(\mathrm{n}+1)}$ to the head of an n step self avoiding walk implies concatenation of the vector $\mathrm{X}^{(\mathrm{n}+1)}$ to the n step self avoiding walk $(\omega(0), \omega(1), \ldots, \omega(\mathrm{n}))$ after ensuring the self avoidance property is maintained. Appending a step (unit vector) to he head of an n step memory τ walk is analogously defined. Also analogously defined is the appending a square or a hook to the head of an n step memory τ walk.

Appending a step (unit vector) $\mathrm{X}^{(\mathrm{i})}$ to the tail of an n step self avoiding walk implies translation by the vector $\mathrm{X}^{(\mathrm{i})}$ of all the vertices of the n step self avoiding walk $(\omega(0), \omega(1), \ldots, \omega(n))$ renaming the resulting points$(\omega(1), \omega(2), \ldots, \omega(n+1))$ and $\omega(0)$ the origin, after ensuring the self avoidance property is maintained. Appending a step (unit vector) to the tail of an n step memory τ walk is analogously defined. Also analogously defined is the appending a square or a hook to the tail of an n step memory τ walk.

These ideas on appending a step, square or hook to the tail or head of a walk will be used in Theorem 2.1 and Lemma 2.7 in the next section, and a clearer picture of these concepts will emerge on reading Chapter 7 of Ref. [6]

The average distance (squared) from the origin after n steps then given by the mean squared displacement

$$\left\langle |\omega(n)|^2 \right\rangle = \frac{1}{C_n} \sum_{\omega:|\omega|=n} |\omega(n)|^2$$

Given two sites x and y let $C_n(x, y)$ be the number of n step SAW ω with $\omega(0) = x$ and $\omega(\mathrm{n}) = y$. The two point function is the generating function for the sequence $C_N(x, y)$,

$$\text{i.e. } G_z(x, y) = \sum_{n=0}^{\infty} C_n(x, y)\, z^N = \sum_{\omega:x\to y} z^{|\omega|}$$

The following definitions are for Sect. 3 on the lace expansion [1].

Given a walk $\omega = (\omega(0), \omega(1), \ldots, \omega(n))$ and two times s and t in $\{0, 1, 2, \ldots, n\}$, we define

$$U_{st}(\omega) = \begin{cases} -1 & \text{if } \omega(\mathrm{s}) = \omega(t) \\ 0 & \text{if } \omega(\mathrm{s}) \neq \omega(\mathrm{t}) \end{cases}$$

Then the two point function can be written

$$G_z(0, x) = \sum_{\omega:0\to x} z^{|\omega|} \prod_{0\le s<t\le|\omega|} (1 + U_{st}(\omega))$$

In an interval $[a, b]$ we refer to a pair $\{s, t\}(s < t)$ of integers in the interval as an edge.

A set of edges in an interval $[a, b]$ is called a graph. (This definition of a graph is used only in Sect. 3).

Two vertices $s, t\,(s <$ t) of a graph are connected by an edge if $\omega(s) = \omega(t)$ in the corresponding walk.

A graph on the interval $[a, b]$ is connected if both a and b are end points of an edge in the graph and any point in the interval $[a, b]$ is covered by an edge in the graph, i.e. lies in between the end points of an edge in a graph.

A lace is a minimally connected graph, i.e. a connected graph for which the removal of an edge would result in a disconnected graph.

A lace associated with a memory τ walk would have all edges of length less than or equal to τ.

Each graph has a unique lace associated with it [6].

Given a connected graph λ, we define the lace L_λ associated with λ as follows:

L_λ consists of edges $s_1t_1, s_2t_2, \ldots, s_it_i$ where

$$s_1 = a, t_1 = \max\{t : at\, \varepsilon\, \lambda\}$$

$$t_i = \max\{t : st\, \varepsilon\, \lambda, s < t_{i-1}\}$$

$$s_{i-1} = \min\{s : st_{i-1} \varepsilon \lambda\}$$

Edges in a graph not in the lace associated with it are said to be compatible with the lace.

We follow Bondy and Murthy [11] (Chapter 6) for terminology of search trees. Our search trees are rooted trees (root defined as $\boldsymbol{r}$) defined by Algorithm 1 (Depth first Search) and Algorithm 2 (Breadth first search) of the next section. A leaf of a tree is a vertex of degree one. The level of a vertex v in a tree T is the length of the path $\boldsymbol{r}$ Tv.

2 Depth first search trees and breadth first search trees

We describe below the Algorithm for Self avoiding walks using the Depth first search tree method, followed by a Table showing the output:

Algorithm 1 (Depth first Search)

In what follows:

If $-\varepsilon_1, \varepsilon_1, -\varepsilon_2, \varepsilon_2, \ldots, -\varepsilon_d, \varepsilon_d$ are the $2d$ unit vectors in d dimensional Euclidean space.

$\boldsymbol{F}$ refers to the set $(-\varepsilon_1, \varepsilon_1, -\varepsilon_2, \varepsilon_2, \ldots, -\varepsilon_d, \varepsilon_d)$, accessible in that order. $l\,(v)$ refers to the level of the vertex $\boldsymbol{v}$., $t\,(v)$ refers to the time of visit to a vertex $\boldsymbol{v}$, *first* ($\boldsymbol{v}$) refers to the time of the first visit to $\boldsymbol{v}$, *last*($\boldsymbol{v}$) refers to the time of the second (last) visit to $\boldsymbol{v}$.

C refers to the family of n step self avoiding walks. This might refer to the co ordinates of the entire walk, or the moments of the end to end distance, radius of gyration or any such relevant parameter.

St refers to the Stack used in the algorithm.

Set the vertex $\boldsymbol{v} := \boldsymbol{u}$ refers to the co ordinates of the self avoiding walk resulting from appending the vector $\boldsymbol{u}$ to the self avoiding walk at the previous level.

Algorithm 1

INPUT: Number of steps *n*, dimension *d*, Family of unit vectors in *d* dimension ***F***.

OUTPUT: Family of self avoiding walks of *n* steps **C**, $\sum_{i=0}^{n} C_i$, time of visit of each vertex of the DFS tree, time of first visit to each vertex of the DFS tree, time of second (last) visit to each vertex of the DFS tree.

```
1:  Set $i \Leftrightarrow 0$ and $St \Leftrightarrow \varphi$ .
2:  Set the origin as the root of the tree r and set v:=r.
3:  increment i by 1.
4:  Set $t(v) \Leftrightarrow 1$, first(v) := t(v), and $l(v) \Leftrightarrow 0$.
5:  Place v at the top of the stack St .
6:  for j:=1 to j:=N do set E(j) := F
7:    while St is non empty do
8:      if l(v) <= n do
9:      set k:=1
10:           if E(k) is non empty do
11:           Consider the head u of E(k) and set v:=u
12:           Place v at the top of the stack St and label the resulting walk as x
13:                if x is self avoiding do
14:                remove u from E(k)
15:                set l(v) := k
16:                increment k by 1 and i by 1.
17:                set t(v) := i , and first(v) := t(v)
18:                end if
19:                else remove u from E(k)
20:           end if
21:           else
22:            set E(k) := F
23:           remove v from the top of the stack St
24:           decrement k by 1 and increment i by 1
25:           set t(v) := i and last(v) := t(v)
```

```
26:                 **end else**
27:         **end if**
28:         **else**
29:         increment $i$ by 1
30:         set $t(\mathbf{v})$ := i and $last(\mathbf{v})$ := $t(\mathbf{v})$
31:          append $x$ to **C**
32:         remove **v** from the top of the stack $St$
33:         decrement $k$ by 1
34          **end else**
35:     **end while**
36:  return (**C**,$t$,$first$,$last$)
```

2.1 Computer implementation of the depth first search algorithm

In the Table given below we display the computer output of the DFS algorithm for four step self avoiding walks in two dimensions. The first number is chronological order in which a vertex appears, 1 represents (0, 0), 2 represents (−1, 0), 3 represents (−2, 0) etc. The second number represents the time of first visit to a vertex and the third number the time of last (second) visit to a vertex. The third number is 0 if the vertex has not been visited a second time. To explain further the first walk to be stored is [(0, 0), (−1, 0), (−2, 0), (−3, 0), (−4, 0)] and this is given by:

1 1 0, 2 2 0, 3 3 0, 4 4 0, 5 5 6

The second walk to be stored is [(0, 0), (−1, 0), (−2, 0), (−3, 0), (−3, −1)] and this is given by:

1 1 0, 2 2 0, 3 3 0, 4 4 0, 6 7 8

Self Intersection occurs after: 23 41 42

The number of vertices in the tree is 153. There is one vertex at level 0 (the root), 4 vertices at level 1 (the number of single step self avoiding walks), 12 vertices at level 2 (the number two step self avoiding walks), 36 vertices at level 3 (the number of 3 step self avoiding walks) and 100 vertices at level 4 (the number of 4 step self avoiding walks).

4 step self Intersection occurs after: [23, 41, 42], [34, 63, 64], [57, 109, 110], [68, 131, 132], [89, 173, 174], [100, 196, 201], [123, 241, 242] and [134, 264, 269]

It would be possible to draw the entire DFS tree using the data above. For example if one chooses [1, 1, 306] followed by [2, 2, 77], [3, 3, 28], [4, 4, 11] then the leaves of the tree would be [5, 5, 6], [6, 7, 8] and [7, 9, 10]. Similarly if we choose [1, 1, 306] followed by [2, 2, 77], [16, 29, 52], [21, 38, 43] then the leaves of the tree would be [22, 39, 40] and [23, 41, 42] the node after [23, 41, 42] being deleted as a result of a 4 step intersection.

Algorithm 2 (Breadth first Search)

Here **Q** refers to the queue used in the Algorithm. **C** and ***F*** are the same as in Algorithm 1.

INPUT: Number of steps *n*, dimension *d*, Family of unit vectors in *d* **dimensions *F***.

OUTPUT: Family of self avoiding walks of *n* steps **C**, $\sum_{i=0}^{n} c_i$.

```
1:  Set $i \Leftrightarrow 0$ and $Q \Leftrightarrow \varphi$ .
2:  Set the origin as the root of the tree r.
3:  increment i by 1.
4:  Set $t(r) \Leftrightarrow i$ and $l(\mathbf{r}) \Leftrightarrow 0$.
5:  Append r to Q.
6:  while Q is non empty do
7:      Set $\boldsymbol{E} := \boldsymbol{F}$
8:      Consider the head x of Q.     /* x is an l(x) step self avoiding walk.
9:      if l(x) < n then
10:         if E is non empty then
11:         append the top most element of E to x and label the resulting walk as y
12:          remove the top most element of E.
12:             if y is self avoiding then
13:             increment i by 1.
14:             Set $t(y) \Leftrightarrow i$ and $l(y) \Leftrightarrow l(x)+1$.
15:             Append y to Q.
16:             end if.
17:             else go to 10
18:           end if
19:           else remove x from Q and Set $\boldsymbol{E} := \boldsymbol{F}$.
20:         end if
21:         else append x to C and remove x from Q.
22:     end while.
23:     return (C,t).
```

Table 1 Nodes of the 4 step self avoiding walk tree in two dimensions

Level 0 (Root)	[1 1 306]
Level 1	[2 2 77] [40 78 153] [78 154 229] [116 230 305]
Level 2	[3 3 28] [16 29 52] [28 53 76] [41 79 104] [54 105 128] [66 129 152] [79 155 178] [91 179 202] [103 203 228] [117 231 254] [129 255 278] [141 279 304]
Level 3	[4 4 11] [8 12 19] [12 20 27] [17 30 37] [21 38 43] [24 44 51] [29 54 61] [33 62 67] [36 68 75] [42 80 87] [46 88 95] [50 96 103] [55 106 111] [58 112 119] [62 120 127] [67 130 135] [70 136 143] [74 144 151] [80 156 163] [84 164 171] [88 172 177] [92 180 187] [96 188 195] [100 196 201] [104 204 211] [108 212 219] [112 220 227] [118 232 239] [122 240 245] [125 246 253] [130 256 263] [134 264 269] [137 270 277] [142 280 287] [146 288 295] [150 296 303]
Level 4	[5 5 6] [6 7 8] [7 9 10] [9 13 14] [10 15 16] [11 17 18][13 21 22] [14 23 24] [15 25 26] [18 31 32] [19 33 34] [20 35 36] [22 39 40] [23 41 42] [25 45 46] [26 47 48] [27 49 50] [30 55 56] [31 57 58] [32 59 60] [34 63 64] [35 65 66][37 69 70] [38 71 72] [39 73 74] [43 81 82] [44 83 84] [45 85 86] [47 89 90] [48 91 92] [49 93 94] [51 97 98] [52 99 100] [53 101 102] [56 107 108] [57 109 110] [59 113 114] [60 115 116] [61 117 118] [63 121 122] [64 123 124] [65 125 126] [68 131 132][69 133 134] [71 137 138] [72 139 140] [73 141 142] [75 145 146] [76 147 148] [77 149 150] [81 157 158] [82 159 160] [83 161 162] [85 165 166] [86 167 168] [87 169 170] [89 173 174] [90 175 176] [93 181 182] [94 183 184] [95 185 186] [97 189 190] [98 191 192] [99 193 194] [101 197 198] [102 199 200] [105 205 206] [106 207 208] [107 209 210] [109 213 214] [110 215 216] [111 217 218] [113 221 222] [114 223 224] [115 225 226] [119 233 234] [120 235 236] [121 237 238] [123 241 242] [124 243 244] [126 247 248] [127 249 250] [128 251 252] [131 257 258] [132 259 260] [133 261 262] [135 265 266] [136 267 268] [138 271 272] [139 273 274] [140 275 276] [143 281 282] [144 283 284] [145 285 286] [147 289 290] [148 291 292] [149 293 294] [151 297 298] [152 299 300] [153 301 302]

2.2 Trees for self avoiding walks

Lemma 2.1 *The B.F.S. and D.F.S. trees are isomorphic.*

Lemma 2.2 *The number of vertices at the mth level of the tree is C_m.*

Lemma 2.3 *In the D.F.S. tree last $(\boldsymbol{r}) = 2\sum_{i=0}^{n} C_i$.*

Proof *The* Depth first tree visits each vertex twice. □

In the D.F.S. tree if we label the vertices of the tree as $p_1^0 = \boldsymbol{r}, p_1^1, p_2^1, \ldots p_{2d}^1, \ldots,$ i.e. $p_i^m, i = 1, 2, \ldots C_m$ is the ith point while moving from left to right at the mth level, i varying from 1 to C_m.

Lemma 2.4 *In the D.F.S. tree for an n step self avoiding walk if p_i^m is the kth point to be visited,* (*i.e. k represents the first number in the triplet in Table* 1) *in Algorithm* 1 *then:*

$$\mathit{first}\left(p_i^m\right) = k, if\ k \le n$$
$$\mathit{first}\left(p_i^m\right) = 2k - m - 1, \quad if\ k > n$$

Proof If $k \le n$ the vertex is being visited for the first time. If $k > n$ then *first* $\left(p_i^m\right) = k + k - m - 1$ since $k - m - 1$ points are visited twice. □

2.3 Trees for simple random walks (memory 0 walks)

In Algorithm 1 if line No. 13 and line No. 18 are deleted and line 19 is changed to: remove $\boldsymbol{u}$ from $E(k)$, Lemmas 2.1–2.4 would carry through without any change. Also the following Lemma would hold.

Lemma 2.5 *For simple random walks (memory 0 walks) in the D.F.S tree if we label the vertices according to their time of incorporation, $i = t(\boldsymbol{v})$ in Algorithm 1*

$$\begin{aligned}
&\textit{for any vertex } i \textit{ at the } n\text{th } \textit{level (i.e. a leaf of the tree)}\\
&\quad \mathit{last}\,(i) - \mathit{first}\,(i) = 1\\
&\textit{for any vertex } i \textit{ at a level } < n \textit{ and } \geq 1\\
&\quad \mathit{last}\,(i) - \mathit{first}\,(i) = [2d]\left[\mathit{last}\,(i+1) - \mathit{first}\,(i+1) + 1\right] + 1
\end{aligned} \tag{2.1}$$

Proof The difference between the first and last visit at the leaf of a tree (nth level) is 1.

For any level less than n, $\mathit{first}\,(i+1) = \mathit{first}\,(i) + 1$, and there are $2d$ branches at each vertex, plus 1 for moving from one branch to the next, and finally 1 to revisit vertex i. □

2.4 Trees for random walks with exclusion of immediate reversals (memory 2 walks)

In Algorithm 1 if line 13 is replaced with:

```
13 : if x is a memory 2 walk do
```

then Lemmas 2.1–2.4 would still be valid. Also the following Lemma would hold:

Lemma 2.6 *For a random walk, with exclusion of immediate reversals (i.e. a memory two walk) in the D.F.S tree if we label the vertices according to their time of incorporation, i.e. $i = t(\boldsymbol{v})$ in Algorithm 1.*

$$\begin{aligned}
&\textit{for any vertex } i \textit{ at the } n\text{th } \textit{level (i.e. a leaf of the tree)}\\
&\quad \mathit{last}\,(i) - \mathit{first}\,(i) = 1\\
&\textit{for any vertex } i \textit{ at a level } < n \textit{ and } \geq 1\\
&\quad \mathit{last}\,(i) - \mathit{first}\,(i) = [2d-1]\left[\mathit{last}\,(i+1) - \mathit{first}\,(i+1) + 1\right] + 1
\end{aligned} \tag{2.2}$$

Proof The difference between the first and last visit at the leaf of a tree (nth level) is 1.

For any level less than n, $\mathit{first}\,(i+1) = \mathit{first}\,(i) + 1$, and there are $2d - 1$ branches at each vertex, plus 1 for moving from one branch to the next, and finally 1 to revisit vertex i. □

2.5 Trees for memory 4 walks

In Algorithm 1 if line 13 is replaced with:

```
13: if x is a memory 4 walk do
```

then Lemmas 2.1–2.4 would still be valid.

In the foregoing if C_n denotes the number of n step memory 4 walks, P_n denotes the number of n step memory 4 walks ending in a square and Q_n denotes the number of n step memory 4 walks beginning and ending in a square.

If we compare the DFS tree for memory two walks with the DFS tree for memory 4 walks we obtain:

$$\begin{aligned}
C_4 &= (2d)(2d-1)^3 - (2d)(2d-2)\\
C_5 &= (2d-1)C_4 - (2d-2)(2d)(2d-2)\\
C_6 &= (2d-1)C_5 - (2d-1)(2d-2)(2d)(2d-2)\\
C_7 &= (2d-1)C_6 - (2d-1)(2d-1)(2d-2)(2d)(2d-2) + (2d-3)(2d)(2d-2)
\end{aligned}$$

Theorem 2.1 *For memory 4 walks:*

$$C_n = [2d-1]\left[C_{n-1} - P_{n-1]}\right] + [2d-3]P_{n-3} + P_{n-4}, \quad n \geq 8 \tag{2.3}$$

Proof We obtain an n step memory 4 walk from an $n-1$ step memory 4 walk by appending a step at the tail of walk in $2d-1$ ways and excluding n step walks that end in a square. Hence:

$$C_n = [2d-1]C_{n-1} - P_n \tag{2.4}$$

Next to determine P_n we append a step to the tail of an $n-1$ step walk ending in a square, in $2d-1$ ways, and then exclude walks that begin and end in a square (this is the inclusion exclusion procedure followed by the DFS algorithm). Hence:

$$P_n = [2d-1]P_{n-1} - Q_n \tag{2.5}$$

Now:

$$Q_n = [2d-3]P_{n-3} + P_{n-4} \tag{2.6}$$

To see this:

If $n > 8$ We can append a hook to the tail of an $n-3$ step walk ending in a square in $2d-3$ ways, plus one additional appendage for a walk ending in a square for which the step being appended is along the same direction as the first step (not orthogonal) which contributes the P_{n-4} term.

If $n = 8$ We can append a hook to a 5 step walk ending in a square whose first two steps are in the same direction in $2d-3$ ways. In a similar manner for a 5 step walk ending in a square whose first two steps are orthogonal. The only exception in the case of a 5 step walk ending in a square whose first two steps are orthogonal being the case of the first step and the square are in the *same* plane, whence one additional appendage is for the eight step walk whose first 4 steps are a square, whose last four steps are a square and the second, third, fourth, fifth, sixth and seventh steps result in a six step reversal (which is allowed). □

Next we need to express P_n in terms of of C_i and P_i for $i < n$. This can be done from geometric considerations or by examining the DFS algorithm:

Lemma 2.7

$$P_n = [2d-2]^2 C_{n-4} - [2d-3] P_{n-3} \tag{2.7}$$

Proof Append a square to the head of an $n-4$ step walk with memory 4 and exclude an $n-3$ step walk ending in a square. We can append a square to the head of an $n-4$ step walk with memory 4 in $(2d-2)\,(2d-3) + (2d-2) = (2d-2)^2$ ways. The first term here refers to the $n-4$th step and the $n-3$th step being orthogonal (there are $2d-2$ such cases) and the second term refers to the $n-4$th and $n-3$th being in the same direction. Finally we can exclude $n-3$ step walks ending in a square in $2d-3$ ways. □

Theorem 2.2 *For any* $n > 4$*:*

$$(a)\ C_n = (2d-1)\,C_{n-1} - (2d-3)\,C_{n-3} - C_{n-4} \tag{2.8}$$

$$(b)\ \chi^4(z) = \frac{-[(2d-1)\,z^3 + 2z^2 + 1]}{z^3 + (2d-2)\,z^2 + (2d-2)\,z - 1} \tag{2.9}$$

(c) $1/\mu^4$ *is equal to the smallest root of the cubic equation* :

$$z^3 + (2d-2)\,z^2 + (2d-2)\,z - 1 = 0 \tag{2.10}$$

Proof (a) Equating the RHS of (2.5) after substituting for $\boldsymbol{Q}_\mathrm{n}$ in (2.6) in the proof of Theorem 1 and (2.7) in Lemma 7 we obtain :

$$(2d-2)^2 C_{n-4} = (2d-1)\,P_{n-1} - P_{n-4} \tag{2.11}$$

From (2.4) we have

$$P_n = (2d-1)\,C_{n-1} - C_n \tag{2.12}$$

Applying (2.12) to P_{n-1} and P_{n-4} we get:

$$\begin{aligned}(2d-2)^2 C_{n-4} = (2d-1)^2 C_{n-2} - (2d-1)\,C_{n-1} \\ + C_{n-4} - (2d-1) C_{n-5}\end{aligned} \tag{2.13}$$

Substituting n for $n-1$ and rearranging we get:

$$\begin{aligned}(2d-1)\,C_n - (2d-1)^2 C_{n-1} + (2d-1)\,(2d-3)\,C_{n-3} \\ + (2d-1)\,C_{n-4} = 0\end{aligned} \tag{2.14}$$

which gives (2.8).

(b) $$\chi^4(z) = 1 + (2d)\,z + 2d\,(2d-1)\,z^2 + 2d(2d-1)^2 z^3 + 2d(2d-1)^3 z^4 - 2d\,(2d-2)\,z^4 + (2d-1)\,z\left[\chi(z) - 1 - (2d)\,z - 2d\,(2d-1)\,z^2 - 2d\,(2d-1)^2 z^3\right] - (2d-3)\,z^3\,[\chi(z) - 1 - (2d)\,z] - z^4[\chi(z) - 1]$$

which on rearrangement gives:

$$\chi^4(z) = \frac{-(2d-1)\,z^4 + (2d-3)\,z^3 + z + 1}{z^4 + (2d-3)\,z^3 - (2d-1)\,z + 1} \tag{2.15}$$

On factoring $z - 1$ we get (2.9)

(c) $1/\mu^4$ is the radius of convergence of $\chi(z)$ [1]. □

Note 1 It is well known [6,9] that the connective constant of the memory 4 walk is given by the largest root of the cubic equation:

$$z^3 - (2d-2)\,z^2 - (2d-2)\,z - 1 = 0 \tag{2.16}$$

If for $d = 2, 3, 4, \ldots$ if α, β, and γ are the roots of Eq. (2.16) where α is real, and β, γ are complex and α β γ $= 1$, then $1/\alpha$, $1/\beta$, and $1/\gamma$ are the roots of (2.10).

Note 2 It would have been possible to derive (2.8) directly. The method followed here is to demonstrate the use of DFS trees to deduce this result.

3 The lace expansion

In this section we prove the following Theorem

Theorem 3.1 *For memory 4 walks of n steps, the number of laces* $f(n) = (-1)^{\frac{n-2}{2}}\left(\frac{1}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-4} + \beta^{n-4}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-4} - \beta^{n-4}\right)$ *when n is even and* $f(n) = (-1)^{\frac{n-1}{2}}\left(\frac{2}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-4} + \beta^{n-4}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-4} - \beta^{n-4}\right)$ *when n is odd. Where α and β are the roots of the equation $x^2 - x - 1 = 0$.*

The lace expansion for memory 2 walks is explained in Ref. [1]. If we calculate the number of laces for memory 4 walks we get:

(No. of steps)	(No. of laces)
4	1
5	3
6	5
7	8
8	14
9	24

In calculation of the number of laces for 6 steps we exclude the lace with an edge joining $\{0, 4\}$ and a second edge joining $\{2, 6\}$. The reason is that the walk associated with this is lace would necessarily have a compatible edge either between $\{1, 5\}$ or $\{3, 5\}$. This reasoning would be applicable for all laces representing walks with number of steps greater than 6.

Let us arrange the memory four walk as follows. Let n_i be the number of laces that end in $\{i, i+4\}$. Clearly such laces do not contain edges that begin from the second point of other edges of length four of the lace. Now each lace in $n_0, n_1, n_2, \ldots\ldots, n_{i-3}$ can be extended to a lace in n_i by the addition of edge $\{i, i+4\}$. But no lace from n_{i-2} can be extended to a lace in n_i because i is the mid-point of the edge $\{i-2, i+2\}$ which is present in the lace of n_{i-2}. Also those laces in n_{i-1} which do not contain the edge $\{i-2, i+2\}$ can be extended to a lace in n_i. Let the number of such laces be n^*_{i-1}.

The lace containing the edge $\{i, i+4\}$ with all other edges are of length two is also counted in n_i. No other lace can be extended to a lace in n_i. As a result

$$n_i = 1 + n_0 + n_1 + \cdots + n_{i-3} + n^*_{i-1}. \tag{3.1}$$

From the extension technique which is discussed above it follows that

$$\begin{aligned} n^*_{i-1} &= n_{i-1} - n^*_{i-2} \\ &= n_{i-1} - \left(n_{i-2} - n^*_{i-3}\right) \\ &= n_{i-1} - n_{i-2} + \left(n_{i-3} - n^*_{i-4}\right) \\ &= \ldots\ldots\ldots\ldots\ldots\ldots \\ &= n_{i-1} - n_{i-2} + n_{i-3} - n_{i-4} + \cdots \pm n_0 \end{aligned}$$

Therefore from (3.1),

$$n_i = n_{i-1} - n_{i-2} + 2\,(n_{i-3} + n_{i-5} + n_{i-7} + \cdots) + 1 \tag{3.2}$$

Replacing i by $i-2$,

$$n_{i-2} = n_{i-3} - n_{i-4} + 2\,(n_{i-5} + n_{i-7} + n_{i-9} + \cdots) + 1$$

Therefore

$$\begin{aligned} & n_i - n_{i-2} = n_{i-1} - n_{i-2} - n_{i-3} + n_{i-4} + 2n_{i-3} \\ & i.e., n_i - n_{i-2} = n_{i-1} - n_{i-2} + n_{i-3} + n_{i-4} \end{aligned}$$

Therefore we obtain the following Lemma.

Lemma 3.1 *If n_i is the number of laces that end in $\{i, i+4\}$ then the following recursion formula holds:*

$$n_i = n_{i-1} + n_{i-3} + n_{i-4} \tag{3.3}$$

Now by the addition of edges of length 2, each lace counted in $n_0, n_1, \ldots\ldots\ldots\ldots n_r$ can be uniquely extended to a lace on $0, 1, 2, \ldots\ldots r+4$.

Therefore the number of such laces

$$f(r+4) = n_0 + n_1 + \cdots + n_r$$

We have

$$\begin{aligned} f(0) &= f(1) = f(2) = f(3) = 0 \\ f(4) &= 1 \\ n_0 &= 1 \\ n_1 &= 2 = n_0 + 1 \\ n_2 &= 2 = n_1 \end{aligned}$$

From (3.3) we have the following:

$$\begin{aligned} n_i &= n_{i-1} + n_{i-3} + n_{i-4} \\ n_{i-1} &= n_{i-2} + n_{i-4} + n_{i-5} \\ n_{i-2} &= n_{i-3} + n_{i-5} + n_{i-6} \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ n_4 &= n_3 + n_1 + n_0 \\ n_3 &= n_2 + n_0 \\ n_2 &= n_1 \\ n_1 &= n_0 + 1 \\ n_0 &= 1 \end{aligned}$$

Adding all these we have

$$f(i) = f(i-1) + f(i-3) + f(i-4) + 2 \tag{3.4}$$

where $i > 4$ and $f(0) = f(1) = f(2) = f(3) = 0$, $f(4) = 1$.

We restate this result separately as a Lemma.

Lemma 3.2 *If $f(r)$ is the number of lace on $0, 1, 2, 3, \ldots\ldots\ldots r+4$, then the following recursion formula holds:*

$$f(r) = f(r-1) + f(r-3) + f(r-4) + 2$$

where $r > 4$ and $f(0) = f(1) = f(2) = f(3) = 0$, $f(4) = 1$.

Thus:

$$\begin{aligned} f(5) &= 3 \\ f(6) &= 5 \text{ etc.} \end{aligned}$$

Let us now solve the recurrence relation

$$u_n = u_{n-1} + u_{n-3} + u_{n-4} + 2$$

where n $= i - 1$ of (3.4) and is defined for $i > 1$ and

$$\begin{aligned} u_0 &= 0 \\ u_1 &= 0 \\ u_2 &= 0 \\ u_3 &= 1 \end{aligned}$$

Then,

Let

$$U(z) = \sum_{n=0}^{\infty} u_n z^n.$$

Then

$$U(z) = u_0 + u_1 z + u_2 z^2 + u_3 z^3 + \sum_{n=4}^{\infty} u_n z^n. \tag{3.5}$$

Now

$$\begin{aligned} \sum_{n=4}^{\infty} u_n z^n &= \sum_{n=4}^{\infty} (u_{n-1} + u_{n-3} + u_{n-4} + 2) z^n \\ &= z \sum_{n=4}^{\infty} (u_{n-1}) z^{n-1} + z^3 \sum_{n=4}^{\infty} (u_{n-3}) z^{n-3} \\ &\quad + z^4 \sum_{n=4}^{\infty} (u_{n-4}) z^{n-4} + 2z^4 \sum_{n=4}^{\infty} z^{n-4} \\ &= z \left\{ U(z) - u_0 - u_1 z - u_2 z^2 \right\} + z^3 \{ U(z) - u_0 \} \\ &\quad + z^4 U(z) + 2z^4 \frac{1}{1-z}. \end{aligned}$$

Therefore from (3.4),

$$U(z) = z^3 + zU(z) + z^3 U(z) + z^4 U(z) + 2z^4 \frac{1}{1-z}$$

Therefore

$$\begin{aligned}
U(z) &= \frac{z^3+z^4}{(1-z)\left(1-z-z^3-z^4\right)} \\
&= \frac{z^3+z^4}{(1-z)\left(1+z^2\right)\left(1-z-z^2\right)} \\
&= z^3\left\{\frac{2z-1}{5\left(1+z^2\right)} - \frac{1}{1-z} + \frac{7z+11}{5\left(1-z-z^2\right)}\right\} \\
&= z^3\left\{\frac{2z-1}{5\left(1+z^2\right)} - \frac{1}{1-z} + \left(\frac{11+5\sqrt{5}}{10}\right)\frac{1}{1-\alpha z}\right. \\
&\quad \left. + \left(\frac{11-5\sqrt{5}}{10}\right)\frac{1}{1-\beta z}\right\}
\end{aligned}$$

where

$$\begin{aligned}
\alpha &= \frac{1+\sqrt{5}}{2}, \quad \beta = \frac{1-\sqrt{5}}{2}. \\
&= z^3\left\{\left(\frac{2z-1}{5}\right)\left(1-z^2+z^4-z^6+\cdots\right) - \left(1+z+z^2+z^3+\cdots\right)\right. \\
&\quad + \left(\frac{11+5\sqrt{5}}{10}\right)\left(1+\alpha z+\alpha^2z^2+\cdots\right) \\
&\quad \left. + \left(\frac{11-5\sqrt{5}}{10}\right)\left(1+\beta z+\beta^2z^2+\cdots\right)\right\}
\end{aligned}$$

In this expansion the coefficient of z^n is u_n.

Therefore when n is odd,

$$\begin{aligned}
u_n &= \left(\frac{-1}{5}\right)(-1)^{\frac{n-3}{2}} - 1 + \left(\frac{11+5\sqrt{5}}{10}\right)\alpha^{n-3} + \left(\frac{11-5\sqrt{5}}{10}\right)\beta^{n-3} \\
&= (-1)^{\frac{n-1}{2}}\left(\frac{1}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-3}+\beta^{n-3}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-3}-\beta^{n-3}\right)
\end{aligned}$$

When n is even,

$$\begin{aligned}
u_n &= (-1)^{\frac{n-4}{2}}\left(\frac{2}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-3}+\beta^{n-3}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-3}-\beta^{n-3}\right) \\
&= (-1)^{\frac{n}{2}}\left(\frac{2}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-3}+\beta^{n-3}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-3}-\beta^{n-3}\right)
\end{aligned}$$

Now $f(n) = u_{n-1}$.

We hence prove Theorem 3.1.

Proof of Theorem 3.1 When n is even

$$f(n) = (-1)^{\frac{n-2}{2}}\left(\frac{1}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-4} + \beta^{n-4}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-4} - \beta^{n-4}\right)$$

and when n is odd,

$$f(n) = (-1)^{\frac{n-1}{2}}\left(\frac{2}{5}\right) - 1 + \frac{11}{10}\left(\alpha^{n-4} + \beta^{n-4}\right) + \frac{\sqrt{5}}{2}\left(\alpha^{n-4} - \beta^{n-4}\right)$$

where α and β are the roots of the equation $x^2 - x - 1 = 0$.

Acknowledgments The authors would like to thank Prof. K.N.R Nair, Professor (retired), School of Computer Science, Mahatma Gandhi University, Kottayam, Kerala, India, for providing a computer program that calculates the number of self avoiding walks and the sum of the squares of the end to end distance and that uses the Depth first search method. The authors would also like to thank the two advisors to JOMC for their careful reading of the manuscript and many helpful suggestions.

References

1. P.J. Flory, *Statistical Mechanics of Chain Molecules* (Wiley, New York, 1969)
2. I. Nishio, S.T. Sun, G. Swislow, T. Tanaka, Nature **281**, 208–209 (1979). doi:10.1038/281208a0
3. I. Nishio, S.T. Sun, G. Swislow, T. Tanaka, Nature **300**, 243–244 (1982). doi:10.1038/300283a0
4. S. Nisha, Molecular Modelling of Dendrimers. Ph.D. Thesis Mahatma Gandhi University (2013)
5. S. Nisha, A.S. Padmanabhan, The statistical mechanics of growing polymer chains. J. Chem. Phys. (in preparation)
6. N.N. Madras, G. Slade, *The Self-avoiding Walk* (Birkhauser, Boston, 1993)
7. A.J. Guttmann, I.G. Enting, Solvability of some statistical mechanical systems. Phys. Rev. Lett. **76**(3), 344–347 (1996)
8. A.J. Guttmann, Indicators for solvability of lattice models. Discrete Math. **217**, 167–189 (2000)
9. M.E. Fisher, M.F. Sykes, Excluded volume problem and the Ising model of ferromagnetism. Phys. Rev. **114**(1), 45–58 (1958)
10. F.T. Wall, R.A. White, Macromolecular configurations simulated by random walks with limited order of non-self intersections. J. Chem. Phys. **65**, 808–812 (1976)
11. J.A. Bondy, U.S.R. Murthy, *Graph Theory* (Graduate Texts in Mathematics) (Springer, Berlin, 2008)
12. S.E. Alm, Upper bounds for the connective constant of self-avoiding walks. Comb. Probab. Comput. **2**, 115–136 (1993)
13. N. Cisby, (Preprint, 8 Feb 2013). arXiv:1302.2106v1 [cond-mat, stat-mech]
14. A.S. Padmanabhan, Random walks strictly confined to a subspace. Stat. Probab. Lett. **53**(2), 299–303 (2001)
15. A.S. Padmanabhan, Restricted random walk numbers of the first and second kind. J. Theor. Comput. Chem. **3**(2), 155–162 (2004)
16. J.W. Lyklema, K. Kremer, J. Phys. A. Math. Gen. **17**, L691–L696 (1984)
17. K. Kremer, J.W. Lyklema, J. Phys. A. Math. Gen. **18**, 1515–1531 (1985)
18. J.W. Lyklema, K. Kremer, J. Phys. A. Math. Gen. **19**, 279–289 (1986)
19. K. Kremer, J.W. Lyklema, Phys. Rev. Lett. **54**(9), 267–269 (1985)
20. I. Majid, N. Jan, A. Coniglio, H.E. Stanley, Phys. Rev. Lett. **52**(15), 1257–1260 (1984)
21. K. Kremer, J.W. Lyklema, Phys. Rev. Lett. **55**(19), 2091–2092 (1985) (this is a comment on Ref. 8 and the response by the authors of Ref. 8)
22. A.S. Padmanabhan, Growing self avoiding walk trees. J. Math. Chem. doi:10.1007/s10910-013-0267-z
23. B. Derrida, H. Saleur, J. Phys. A. Math. Gen. **18**, L1075–L1079 (1985)
24. B. Derida, J. Phys. A. Math. Gen. **14**, L5–L9 (1985)