Non-local Correlation Functions in the Spanning Tree Model near the Boundary

Khaydar Nurligareev (LIPN, Paris-13)

joint with A. Povolotsky (HSE, Moscow)

4 March 2019

Motivation

- Kirchhoff theorem links Spanning Trees, Dimers, Abelian Sandpile Model, Loop-Erased Random Walks and other combinatorial models.
- Correlation functions in these models can be described by Conformal Field Theories in thermodynamical limit.
- Goal: confirm predictions from Conformal Field Theories using combinatorial methods.

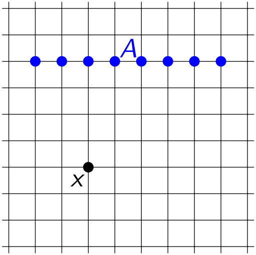
Stating the problem

- Let $\mathcal{G} = (V, E)$ be an undirected connected graph with neither loops nor multiple edges.
- Take two non-intersecting subsets of vertices $I_k = \{i_1, \ldots, i_k\} \subset V$ and $J_k = \{j_1, \ldots, j_k\} \subset V$.
- Consider k totally disjoint loopless paths from I_k to J_k (such configurations of paths are called *watermelons*).
- Typical questions.
  What is the number of different watermelon configurations?
  How does it change with the distance r between I_k and J_k?

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

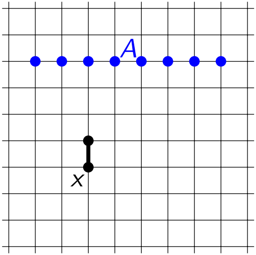
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\qquad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

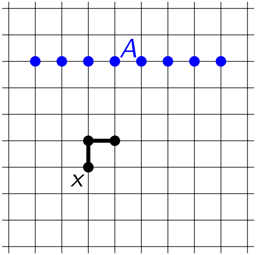
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\qquad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

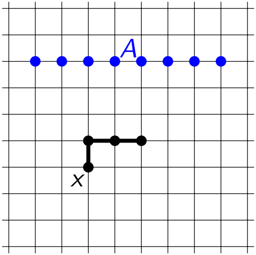
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

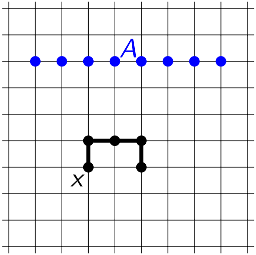
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

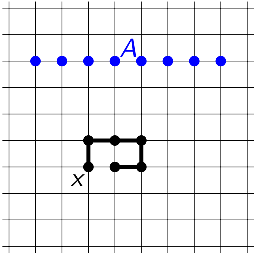
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

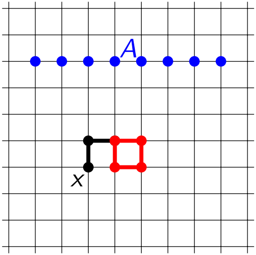
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

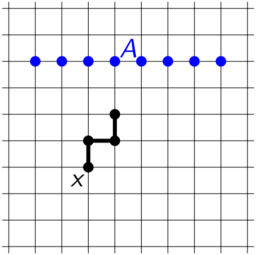
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

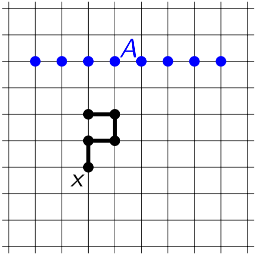
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

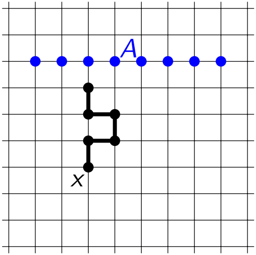
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\qquad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

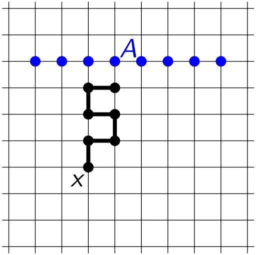
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \dots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

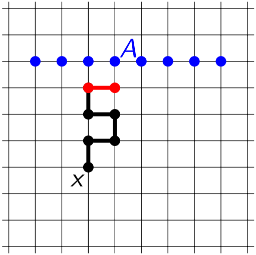
$$LERW(x, A) = (y_0, \dots, y_m) = (X_{n_0}, \dots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



Define *loop-erased random walk* as a path

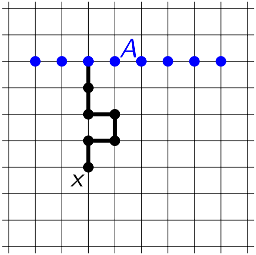
$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Loop-erased random walk (LERW)

Let

- $A \subset V$ be a set of vertices,
- X_n be a simple random walk starting at $X_0 = x$,
- $\tau_A = \min\{n \geqslant 0 \colon \xi_n \in A\}$ be a stopping time (hitting time for the set A),
- $\gamma = (X_0, X_1, \ldots, X_{\tau_A})$ be a path corresponding to X_n.



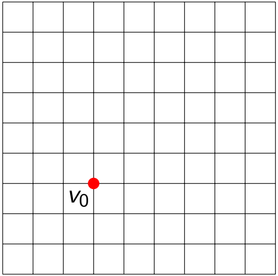
Define *loop-erased random walk* as a path

$$LERW(x, A) = (y_0, \ldots, y_m) = (X_{n_0}, \ldots, X_{n_m}),$$

where $\quad n_0 = 0, \qquad n_{i+1} = \max\{j \colon \gamma(j) = \gamma(n_i)\} + 1.$

Wilson algorithm for generating uniform spanning tree

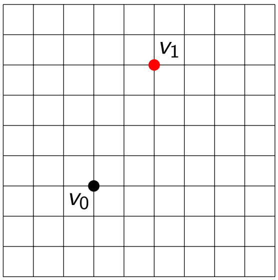
1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. ...
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

Wilson algorithm for generating uniform spanning tree

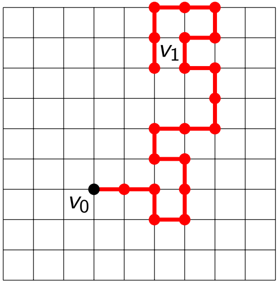
1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. $\ldots$
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

Wilson algorithm for generating uniform spanning tree

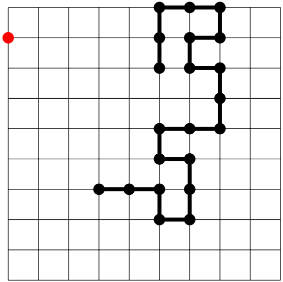
1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. ...
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

Wilson algorithm for generating uniform spanning tree

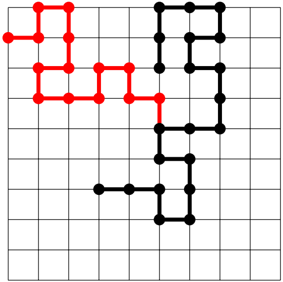
1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. ...
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

Wilson algorithm for generating uniform spanning tree

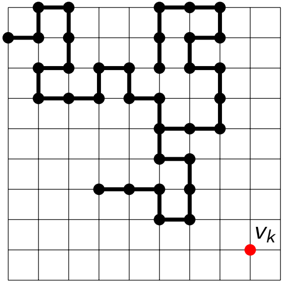
1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. ...
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

Wilson algorithm for generating uniform spanning tree

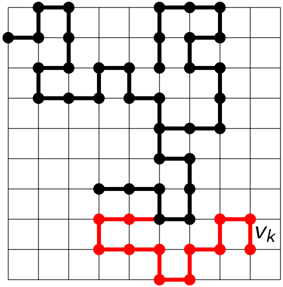
1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. $\ldots$
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

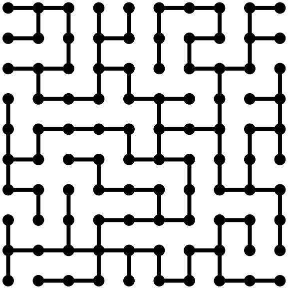
Wilson algorithm for generating uniform spanning tree

1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. ...
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.



If $|V_0| > 0$, then we will get a spanning forest.

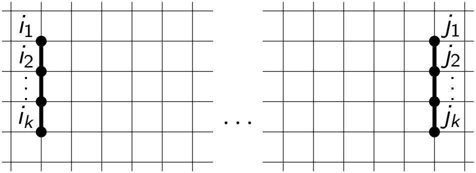
Wilson algorithm for generating uniform spanning tree



1. Take any vertex $v_0 \in V$.
2. Define $\mathcal{U}_0 = \{v_0\}$.
3. Take any vertex $v_1 \in V \setminus \mathcal{U}_0$.
4. Consider $LERW(v_1, \mathcal{U}_0)$ and define $\mathcal{U}_1 = LERW(v_1, \mathcal{U}_0)$.
5. $\ldots$
6. Take any vertex $v_k \in V \setminus \mathcal{U}_{k-1}$.
7. Define $\mathcal{U}_k = LERW(v_k, \mathcal{U}_{k-1}) \cup \mathcal{U}_{k-1}$.
8. At the end, we obtain a spanning tree.

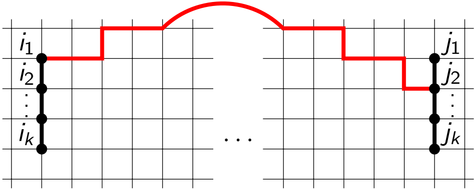
If $|V_0| > 0$, then we will get a spanning forest.

LERW and watermelons



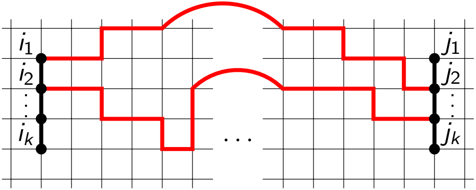
Each k-leg watermelon can be considered as k loop-erased random walks from I_k to J_k.

LERW and watermelons



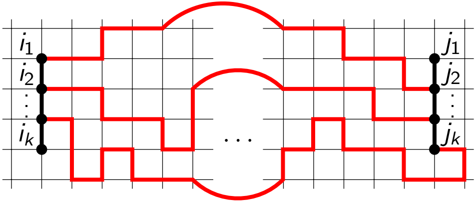
Each k-leg watermelon can be considered as k loop-erased random walks from I_k to J_k.

LERW and watermelons



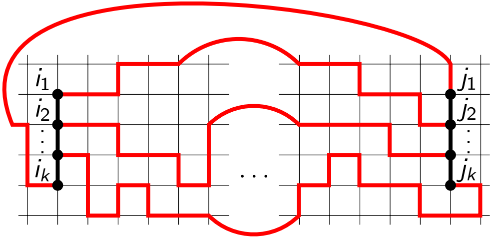
Each k-leg watermelon can be considered as k loop-erased random walks from I_k to J_k.

LERW and watermelons



Each k-leg watermelon can be considered as k loop-erased random walks from I_k to J_k.

LERW and watermelons

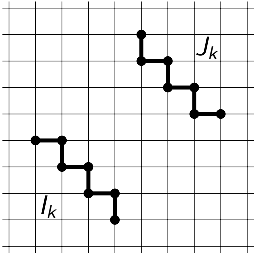


Each k-leg watermelon can be considered as k loop-erased random walks from I_k to J_k.

Previous results: isotropic case

A. Gorsky, S. Nechaev, V. Poghosyan and V. Priezzhev studied this problem in 2013 for the following case:

- $\mathcal{G}$ is a square lattice,
- k is odd,
- I_k and J_k have the form of *fence*.



Let r be the distance between I_k and J_k,

$$W_n(k,r) = \frac{\#\{\text{watermelons inside the square } n \times n\}}{\#\{\text{spanning trees inside the square } n \times n\}},$$

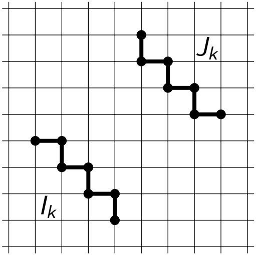
$$W(k,r) = \lim_{n\to\infty} W_n(k,r).$$

<u>Theorem</u>: $W(k,r) \sim C \cdot r^{-\nu} \cdot \ln r,$ where $\nu = \dfrac{k^2-1}{2}$.

Previous results: anisotropic case

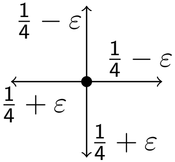
Another case studied by V. Priezzhev, A. Gorsky, S. Nechaev and V. Poghosyan is the following:

- $\mathcal{G}$ is an elongated square lattice,
- k is odd,
- I_k and J_k have the form of *fence*.



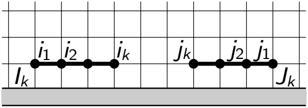
<u>Theorem</u>: if $\varepsilon \to 0$, then

$$W(k,r) \sim C \cdot r^{-\nu}, \quad \text{where } \nu = \frac{k^2}{2}.$$



New results: isotropic case, open boundary

- $\mathcal{G}$ is a square lattice on the half-plane,
- I_k and J_k have the form of segments located near the boundary,
- absorbing boundary conditions.

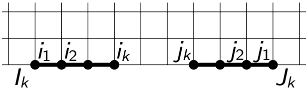


Theorem 1 (N., Povolotsky):

$$W^{op}(I_k, J_k) \sim C^{op} \cdot r^{-k(k+1)}, \quad \text{where} \quad C^{op} = \frac{1}{\pi^k \cdot k!} \cdot \prod_{s=1}^{k} (s!)^2.$$

New results: isotropic case, closed boundary

- $\mathcal{G}$ is a square lattice on the half-plane,
- I_k and J_k have the form of segments located near the boundary,
- reflecting boundary conditions.

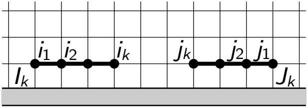


Theorem 2 (N., Povolotsky):

$$W^{cl}(I_k, J_k) \sim C^{cl} \cdot r^{-k(k-1)} \cdot \ln r, \quad \text{where} \quad C^{cl} = \frac{1}{\pi^k \cdot (k-1)!} \cdot \prod_{s=1}^{k} (s!)^2.$$

New results: anisotropic case, open boundary

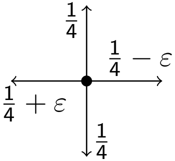
- $\mathcal{G}$ is a horizontally elongated square lattice on the half-plane,
- I_k and J_k have the form of segments located near the boundary,
- absorbing boundary conditions.



Theorem 3 (N., Povolotsky): if $\varepsilon \to 0$, then

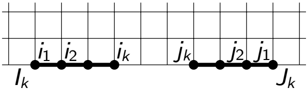
$$W^{op}(I_k, J_k) \sim C^{op} \cdot r^{-k\left(k+\frac{1}{2}\right)},$$

where $$C^{op} = \frac{1}{\sqrt{2^{k^2} \cdot \pi^k}} \cdot \prod_{s=1}^{k-1} s! \cdot (2s+1)!!.$$



New results: anisotropic case, closed boundary

- $\mathcal{G}$ is a horizontally elongated square lattice on the half-plane,
- I_k and J_k have the form of segments located near the boundary,

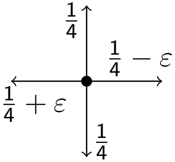


- reflecting boundary conditions.

Theorem 4 (N., Povolotsky): if $\varepsilon \to 0$, then

$$W^{cl}(I_k, J_k) \sim C^{cl} \cdot r^{-k\left(k-\frac{1}{2}\right)},$$

where $$C^{cl} = \frac{1}{\sqrt{2^{k^2} \cdot \pi^k}} \cdot \prod_{s=1}^{k-1} s! \cdot (2s-1)!!.$$



Tools for proof

- Matrix Tree Theorem,
- Green functions,
- Generating functions,
- Symmetric (Schur) functions.

Thank you for your attention!