Mid-Session Report

中間報告プレゼン

Julian M. Rice
ジュリアン・ライス

An Evaluation of the Rotor Router Model: Network Patrolling

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Julian M. Rice
ジュリアン・ライス

Table of Contents

Julian M. Rice
ジュリアン・ライス

Distributed Systems: Basics
分散システム：基本情報

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

What is a *Distributed System*?

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

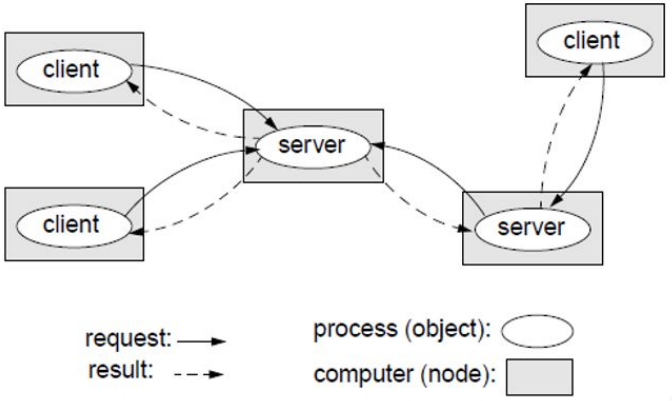
Sources

→ Brooke, P., & Paige, R. (2008). ***Practical Distributed Processing***. Springer.
→ Bharambe A., & Pang, J., & Seshan S. (2006). ***Colyseus: Making distributed FPS games possible***. Carnegie Mellon University.

ENGLISH

A **distributed system** is a system/network of computers autonomously that work together; it requires a method of communication from computer (**node**) to computer and contains **three fundamental characteristics**:

1. **Concurrency**: Multiple processes run at the same time. Think about handling shared access to resources.
2. **Synchronization (in time)**: All actions have to be coordinated and accurate.
3. **Failures**: The system must be designed in a way that if one component fails to run, said system continues to operate.



日本語

分散システムとは「多くのコンピュータが互いに通信して協調動作するシステム」である (増澤ラボ)。分散システムではコンピューターの通信方法が必要であり三つの基本的な特徴がある。

1. **並行性**：複数のプロセスを同時に処理する必要があるので、リソースを割り当てる難易度が高い。
2. **同期**：全ての動作を正確に合わせないといけない
3. **故障**：システムの一部が故障しても正しく動作するシステムを作る必要がある。

Utilization of Mobile Agents

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

→ Brooke, P., & Paige, R. (2008). ***Practical Distributed Processing***. Springer.
→ J. Cao, and S. K. Das, **Mobile Agents in Networking and Distributed Computing**, John Wiley & Sons., 4-16, 2012
→ D. Lange, and M. Ohama, **Seven Good Reasons for Mobile Agents**, Communications of the ACM., 88-89, March 1999

ENGLISH

A **mobile agent** is a bot that contains data and **migrates** from computer to computer within a network; using a single (or multiple) agent(s) can benefit a distributed system because of the following:

1. **Fixing Latency**: "Real-time" changes become more "real" in the sense that latency is decreased.
2. **Reducing Network Load**: Raw data flow can be decreased; agents can transport data to a place where interactions occur locally.
3. **Autonomy**: Agents can work on their own if programmed to do so after being dispatched.
4. **Dynamic Nature**: As the environment changes, agents are able to react autonomously to changes.

日本語

情報、アルゴリズムが入っているボットは **トークン**と呼ばれる。このトークンは別のコンピューターに **移動する**ことが可能。単一 / 複数トークンどちらも以下の理由で分散システムに非常に役に立つ:

1. **レイテンシーを低下させる** : 変化に対する反応速度が向上し、レイテンシーが減少する。
2. **ネットワークフロー減少** :トークンが大切な情報をローカルに処理できる場所まで移動させて、ネットワークフローの容量を空ける。
3. **自律性** : プログラムか命令を出すと、自律的に行動することが可能である。
4. **動的性** : 環境が変わったら、トークンが自律的に変化を認識し、適当な行動を起こす。

Julian M. Rice
ジュリアン・ライス

Rotor Router Model: Basics
ロータールーターモデル：基本情報

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

What is the *Rotor Router Model?*

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

→ V. Yanovski and I. A. Wagner, ***A Distributed Ant Algorithm for Efficiently Patrolling a Network***, *Algorithmica*, 2003

ENGLISH

General model used for figuring out patrolling efficiency problems. What does **patrolling** mean?

Patrolling: Ongoing exploration of a network by a decentralized group of simple memoryless robotic agent(s).

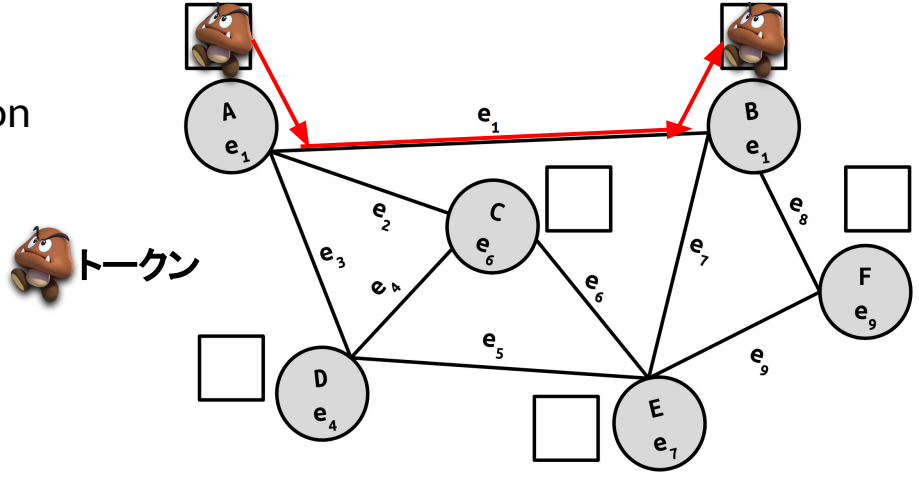
In the paper I have analyzed, the emphasis lies on a **single agent** (one memoryless bot).

The algorithm is not too sophisticated, and it allows for increased patrolling efficiency.

日本語

→ ロータールーターとはネットワークの巡回効率を高めるためのモデル。巡回とは？

→ 私が分析した一つ（最初）の論文ではメモリがない単一トークンの巡回に的を絞っている。



A Few Base Claims... (1/2)

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

→ V. Yanovski and I. A. Wagner, ***A Distributed Ant Algorithm for Efficiently Patrolling a Network***, *Algorithmica*, 2003

ENGLISH

The authors of the paper included some claims about their newly proposed algorithm:

> **[1]** If the graph is Eulerian, then the agent will traverse *all* edges in O(|V||E|).
>
> **[2]** After a **stabilization period**, an **extended Eulerian cycle** is completed.

Stabilization period: An extended period of time where the bot runs on a synchronous network.

Extended Eulerian cycle: A cycle where each edge is traversed the same number of times.

日本語

この論文の著者が新しく作ったアルゴリズムについての簡単な主張とは:

> **[1]** オイラーグラフの場合、トークンは全ての辺をO(|V||E|)以内に巡回する。
>
> **[2]** **長期間**が経ったら、**拡張オイラー閉路**が必ずできる。

長期間:トークンがずっと同期ネットワークで巡回している期間。

拡張オイラー閉路: 全ての辺が同じ回数だけトークンに巡回される閉路。

5 15

A Few Base Claims... (2/2)

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

→ V. Yanovski and I. A. Wagner, ***A Distributed Ant Algorithm for Efficiently Patrolling a Network***, *Algorithmica*, 2003

ENGLISH

The authors of the paper included some claims about their newly proposed algorithm:

[3] Adding more agents **increases traversal frequency**, does *not* increase the time it takes to cover a graph for the first time.

日本語

この論文の著者が新しく作ったアルゴリズムについての簡単な主張とは:

[3] トークンを増やせば、辺の**訪問頻度**が増加するが、オイラーグラフがトークンに初めて完全に巡回される時間は変わらない

	Traversal Frequency 訪問頻度	Graph Cover Time (1st) 完全巡回時間（最初）
Single agent (トークン x1)	↑↑	**T** (*a period*)
Multiple agents (トークン x2+)	↑↑↑↑	**T** (*a period*)

Overview of the Algorithm

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

→ V. Yanovski and I. A. Wagner, ***A Distributed Ant Algorithm for Efficiently Patrolling a Network***, *Algorithmica*, 2003

ENGLISH

Initialization:
for every vertex *v*, set *next exit pointer* in *v* to point to first edge emanating from *v*.

Function:
def RotorRouterAlgorithm(Vertex *u*):
(1) Let *k* be the location of the *next exit pointer* in *u*
(2) *e* = *u* → *v* #e is the corresponding edge
(3) Move *next exit pointer* in *u* to the edge **(k+1) mod degree(u)**
(4) *u* := *v*;
(5) Go to (1)
end RotorRouterAlgorithm

Notes: Vertex *u*, Vertex *v*, Edge *e*, *Next Exit Pointer*, *k*

日本語

初期化
ノード *v* は一番最初に訪問する辺として *next exit pointer* の値を設定する。

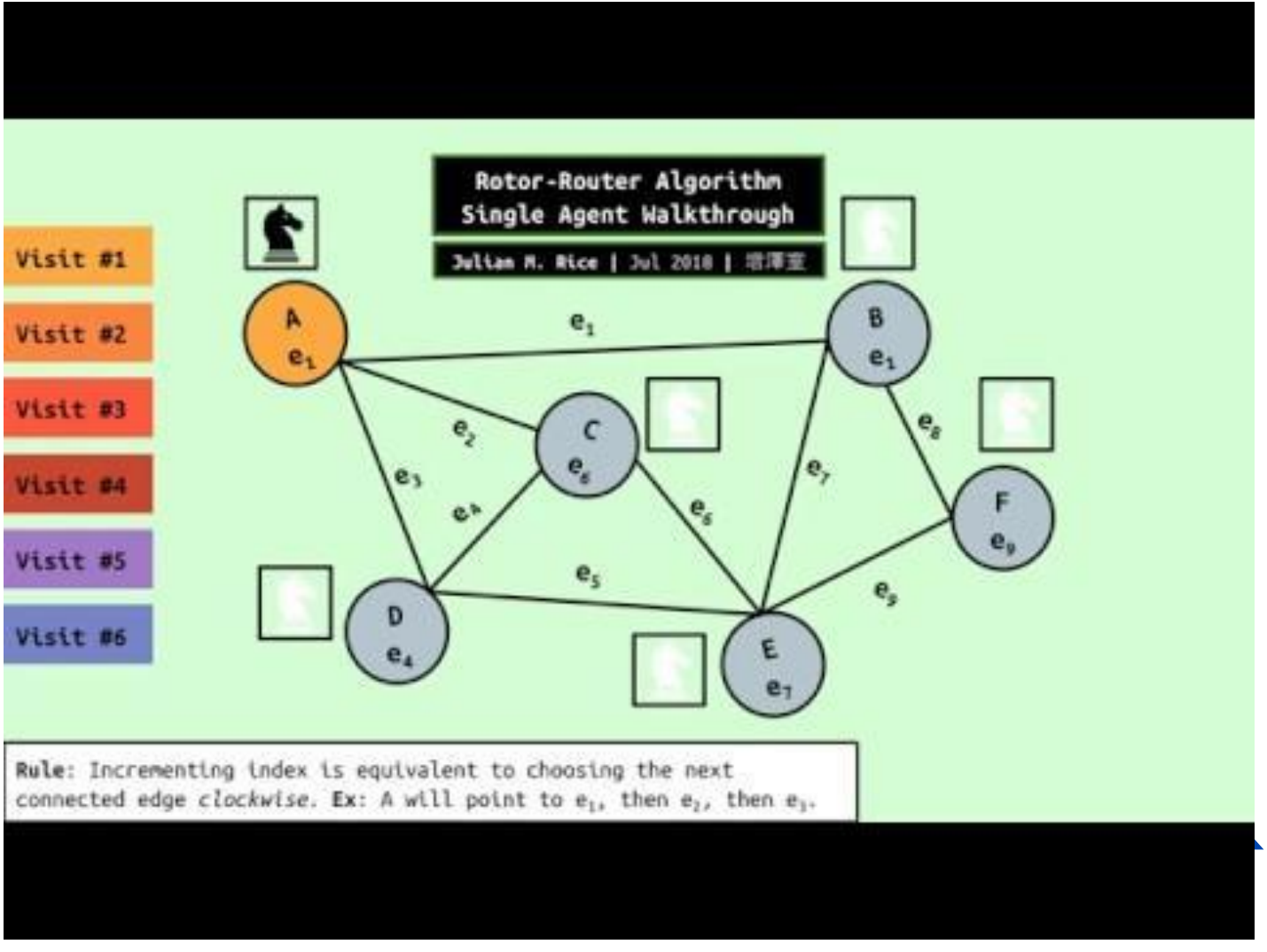
関数
関数 ロータールーター(ノード *u*):
(1) *k* はノード *u* の *next exit pointer* の値
(2) e = u → v #e はノード u とノード v の間の辺
(3) ノード *u* の *next exit pointer* の値は **(k+1) mod degree(u)**
(4) *u* := *v*; #ノードv はノード *u* になる
(5) (1) に戻る
関数 ロータールーター終了

ノート: ノード u, v, 辺 e, *Next Exit Pointer*, *k*

Julian M. Rice
ジュリアン・ライス

Rotor Router Model: Animation
ロータールーターモデル：アニメーション・映像

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018


Rotor-Router Algorithm
Single Agent Walkthrough
Julian M. Rice | Jul 2018 | 培澤至
Visit #1
Visit #2
Visit #3
Visit #4
Visit #5
Visit #6
A
e1
B
e1
C
e6
D
e4
E
e7
F
e9
e1
e2
e3
e4
e5
e6
e7
e8
e9
Rule: Incrementing index is equivalent to choosing the next connected edge clockwise. Ex: A will point to e1, then e2, then e3.

Julian M. Rice
ジュリアン・ライス

Rotor Router Model: Code (inc)
ロータールーターモデル：シミュレーション (まだ不完全)

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Basic Implementation, Thoughts

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

→ Julian! Me!

ENGLISH

Some basic details for what I have produced thus far:

Language: C++

Focus: Object Oriented, Classes

Current (finished) Classes: Edge, Vertex, Graph

Incomplete or in progress details are noted here:

[1] The function for the mobile agent. I have the algorithm and everything at hand; all that remains is for me to actually create it!

[2] Finishing touches: this includes comments, design documentation, polishing variable names, debugging, and lots of example graphs!

日本語

作成中シミュレーションの概要：

C++ 言語

オブジェクト指向なコード、**クラス**の利用

辺、ノード、グラフのクラスを完成させた

まだ開発しているところ、直したいところ：

[1] トークンの関数を作る。アルゴリズムはわかりやすいから、あとは作成するだけだ。

[2] コードを推敲する。コメントを追加し、デザイン資料を作り、デバッグし、そして多様なグラフを使用してシミュレーションする。

```
class Edge {
public:
    Edge(Vertex* start, Vertex* destination);
    void changeStartVertex(Vertex* s)   { m_start = s; }
    void changeEndVertex(Vertex* e)     { m_destination = e; }
    Vertex* getStartVertex()    { return m_start; }
    Vertex* getEndVertex()      { return m_destination; }
private:
    Vertex* m_start;
    Vertex* m_destination;
};
Edge::Edge(Vertex* start, Vertex* destination) {
    m_start = start;
    m_destination = destination;
}
```

Class 1: **Edge** | **辺**

__Vertex__ class is defined before this in a single line

```
class Vertex {
public:
    Vertex(string name, int index, int visitedCount, int edgeCount);
    void addEdge(Vertex* dest);
    void setIndex(int edgeCount);
    void printStatus();
    void printEdges();

    //Accessor Member Functions
    int getIndex() { return m_index; }
    int getVisited() { return m_visitedCount; }
    int getEdgeCount() { return m_edgeCount; }
    string getName() { return m_name; }
private:
    int m_index;
    int m_visitedCount;
    int m_edgeCount;
    string m_name;
    vector<Edge> edges;
};
```

Class 2: Vertex | ノード

The next slide consists of the function definitions

```
Vertex::Vertex(string name, int index, int visitedCount, int edgeCount) {
    m_name = name;
    m_index = index;
    m_visitedCount = visitedCount;
    m_edgeCount = edgeCount;
}

void Vertex::addEdge(Vertex* dest) {
    Edge newEdge(this, dest);
    edges.push_back(newEdge);
    m_edgeCount++;
}

void Vertex::setIndex(int edgeCount) {
    m_index = randomInteger(1, edgeCount);
}
```

Class 2: Vertex | ノード

```
void Vertex::printStatus() {
    string pointingTo = edges[m_index-1].getEndVertex()->getName();
    cout << "----------------------------" << endl;
    cout << "-- VERTEX " << m_name << endl;
    cout << "-- Degree: " << m_edgeCount << endl;
    cout << "-- Times Visited: " << m_visitedCount << endl;
    cout << "-- Pointing to: " << m_index << " (VERTEX " << pointingTo << ")" << endl;
    //cout << "----------------------------" << endl;
}

void Vertex::printEdges() {
    //cout << "----------------------------" << endl;
    cout << "   | Edges of VERTEX " << m_name << " |" << endl;
    for (int i = 0; i < edges.size(); i++) {
        Edge e = edges[i];
        cout << "-- Edge " << i+1 << ": ";
        cout << e.getEndVertex()->getName() << endl;
    }
    cout << endl;
}
```

```
class Graph {
public:
    Graph() {}
    void insertVertex(Vertex* vertex);
    void printGraph();
private:
    vector<Vertex*> vertices;
};

void Graph::insertVertex(Vertex* vertex) {
    vertices.push_back(vertex);
}

void Graph::printGraph() {
    for (int i = 0; i < vertices.size(); i++) {
        vertices[i]->printStatus();
        vertices[i]->printEdges();
    }
}
```

Class 3: Graph | グラフ

Julian M. Rice
ジュリアン・ライス

Japanese: Language & Culture
日本語と日本文化、感想

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

Sources

How is my life in Japan? A Message

→ Julian again

Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

ENGLISH

I learned numerous vocabulary words, worked on onyomi and kunyomi for various kanji, and became more able to express my thoughts throughout my stay in Japan so far! Living on my own is great too!

I felt lonely from time to time, but my friends, significant other, and everyone at Masuzawa's Lab have helped make me feel the spirit of youth!

Other than fun times, I've also gone through insane ones too, which has led to a wider perception of the world. I still have **3** weeks left in Osaka; I'm going to finish these assignments and create unforgettable memories with everyone! Let's go! Kuribo!

日本語

日本に来てから、多様な単語を習って来ました！色々な漢字の音読みと訓読みを学習し、伝えたいことがもっと上手く伝えられるようになりました。一人暮らしも非常に楽しんでいます！

時々寂しくなるかもしれないが、いつも日本人の友達、研究室のみんなといっぱい楽しい経験をしていて、素晴らしい青春を味わっています。

楽しい経験に加えて、たくさん不測の事態（日本人との**戦闘**＋MORE) に陥ったので、世界観が広がって来ました。まだ3週間あるから、一所懸命に論文を書き終わらせながら忘れられない思い出を作りたいと思います！レッツゴー！クリボー！

15 15

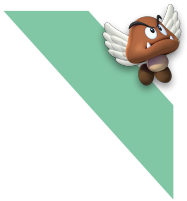
How is my life in Japan? Some Vocab

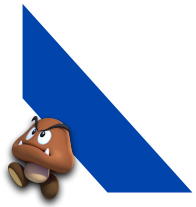
Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018

漢字・単語	英語	漢字・単語	英語
分布（ぶんぷ）	Distribution	基礎（きそ）	Foundation
推定（すいてい）	Estimation	丸出し（まるだし）	Exposed
訓練（くんれん）	Practice, drill	人柄（ひとがら）	Personality
関数（かんすう）	Function	拡張する（かくちょう）	To expand
最尤方（さいゆう方）	Max likelihood estimate	巡回する（じゅんかい）	To go around
野宿（のじゅく）	Sleeping outdoors	動作する（どうさ）	To operate
下心（したごころ）	Ulterior motive	閉路（へいろ）	Closed cycle
美人局（つつもたせ）	英語で難しい...	辺（へん）	Edge

Julian M. Rice
ジュリアン・ライス

Mid-Session Report: End
中間報告プレゼン：終了





Masuzawa Laboratory: *Algorithm Engineering*
増澤研究室 | July 18th, 2018