Fast Integer Linear Programming for Set-Pair Routing Problem

Yasuhiro Takashima
University of Kitakyushu
Kitakyushu, Fukuoka, Japan

Abstract— This paper introduces an efficient approach to integer linear programming for addressing the set-pair routing problem. Unlike previous works, which either employ fast heuristics that may not yield optimal solutions or exact methods that may exceed practical processing time, the proposed method is a rapid integer programming formulation. This approach ensures a balance between practical processing time and the delivery of optimized or high-quality solutions. Empirical evidence validates the efficiency of the proposed method.

I. Introduction

In the domain of LSI physical layout design, addressing the routing problem remains crucial [1–3]. A contemporary challenge arises in FPGA pin routing on printed circuit boards, specifically defined as the set-pair routing problem [3]. This problem gains significance as FPGA pin assignments can be altered after designing the board layout. Consequently, efficient routing solutions are required for arbitrary pairs of pins. Additionally, in high-performance board designs, minimizing signal timing skew is desirable. To tackle this issue, a straightforward model involves minimizing both the maximum length and the differences among various routes.

Existing solutions encompass a variety of methods, including fast heuristics such as those proposed by [4–6], and exact methods utilizing ILP or SAT as seen in [7, 8]. Fast heuristics rely on iterative refinement, achieving rapid convergence but without a guarantee on length performance. On the other hand, exact methods ensure an optimal solution output if the optimum process converges, while the high expense of runtime.

This paper introduces a fast ILP method for practical runtime implementation without compromising optimality. The method streamlines the ILP formulation by eliminating unnecessary variables. Experimental results confirm that the proposed approach provides explicit routing length performance within practical runtime constraints.

II. Set-pair routing problem

The set-pair routing problem, introduced in [3], is defined as follows: Consider a graph $G = (V, E)$, where V and E correspond to the vertex set and the directed edge set, respectively. Additionally, let S and T be the source-pin set and sink-pin set, respectively, both subsets of the vertex set V with a size of m. Connections are established between pins $s \in S$ and pins $t \in T$ over the graph G. Each connection from a source pin s is referred to as the *route* of s. Importantly, each route is not allowed to share any vertices with other routes. The objective of the problem is to minimize the length difference among routes while simultaneously minimizing the maximum length of any route. This problem has been proven to be NP-hard [9].

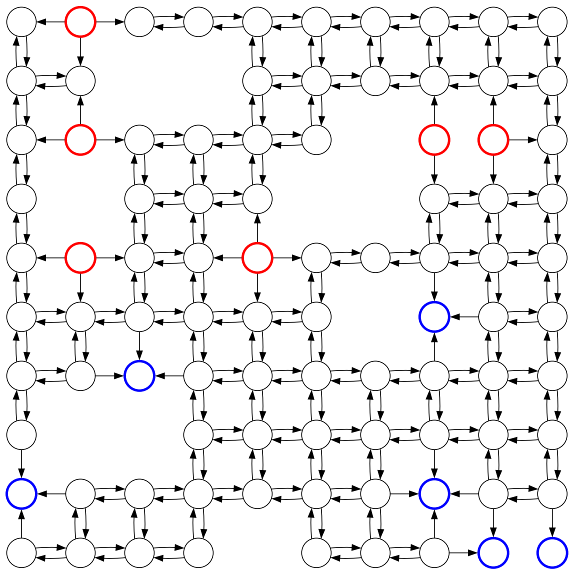
Fig. 1. Example of input

Figures Fig. 1 and Fig. 2 depict the input and output, respectively, of an example illustrating the set-pair routing problem. In these figures, red-colored circles represent the source-pins, blue-colored circles correspond to the sink-pins, and the red-colored lines depict the routes. In the output illustrated in Fig. 2, the maximum length of routes is 12, and the difference in length among the routes is 7.

The solutions for the set-pair routing problem are classified into the following two categories. 1) fast heuristics [4–6], and 2) exact method by ILP or SAT [7, 8].

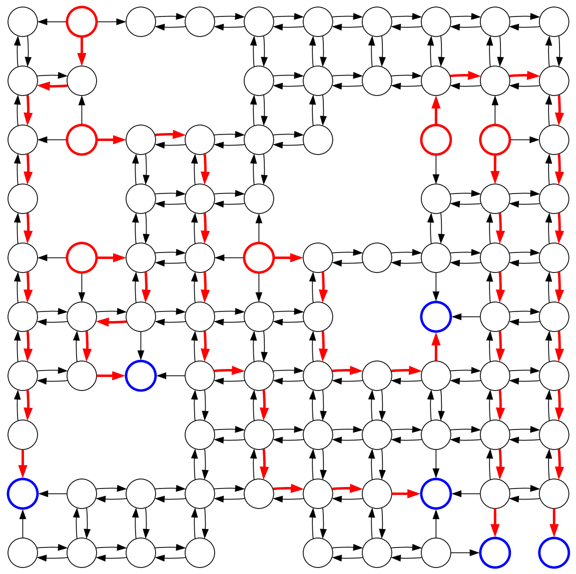

Fig. 2. Example of output

Each fast heuristic in item 1 initiates by constructing a feasible solution. Subsequently, it sequentially refines the preceding solution. Notably, it is observed that the set-pair routing problem, when its objective is transformed into minimizing the total length of all routes, becomes polynomially solvable. Consequently, these fast heuristics leverage total-length minimization to swiftly obtain an initial solution. However, it is crucial to note that these methods, while very fast, lack a guarantee of output quality.

Conversely, for each exact method in item 2, the output must be an optimum solution. Nevertheless, the optimization process comes at a significant cost in terms of processing time and memory usage. This is primarily due to the inclusion of exponential constraints in their formulations, making the optimization process resource-intensive.

III. Proposed method

A. ILP formulation

The proposed method comprises a two-step optimization process. For both optimization steps, the ILP formulation is nearly identical, with the exception of the objective function. The input is specified as follows:

Graph $G = (V, E)$, where Vertex Set $V = \{v_i\}$ and Edge Set $E = \{(v_i, v_j)\}$, Source-pin set $S = \{s_k\} \subset V$, Sink-pin set $T = \{t_h\} \subset V$

Here, the route from source-pin s_k is denoted as r_k.

Next, the variables are defined as follows:

- For route r_k and edge (v_i, v_j), 0-1 variables $x_{k,i,j}$

$$x_{k,i,j} = \begin{cases} 1 & \text{route } r_k \text{ passes through edge } (v_i, v_j) \\ 0 & \text{otherwise} \end{cases}$$

- For route r_k, integer variables z_k, corresponding to the length of r_k

Then, the definition of constraints is outlined below:

Source-pin constraint For the source-pin $s_k (= v_i)$,

- Only route r_k outputs: $\sum_{j|(v_i,v_j)\in E} x_{k,i,j} = 1$
- The other routes do not output: $x_{k',i,j} = 0$ where $k' \neq k, (v_i, v_j) \in E$
- No route inputs: $x_{k,j,i} = 0$ where $\forall k, (v_i, v_j) \in E$

Sink-pin constraint For the sink-pin $t_h (= v_j)$,

- Exact one route inputs: $\sum_k \sum_{i|(v_i,v_j)\in E} x_{k,i,j} = 1$
- No route outputs: $x_{k,j,i} = 0$ where $\forall k, (v_i, v_j) \in E$

Ordinary-pin constraint For not source nor sink pin v_i,

- At most one route outputs: $\sum_k \sum_{j|(v_i,v_j)\in E} x_{k,i,j} \leq 1$
- Out-degree and in-degree are same: $\sum_{l|(v_l,v_i)\in E} x_{k,l,i} - \sum_{j|(v_i,v_j)\in E} x_{k,i,j} = 0$

Route-length constraint The length of route r_k is z_k: $z_k - \sum_{(v_i,v_j)\in E} x_{k,i,j} = 0$

Sub-cycle removal constraint No sub-cycle exists:

$$\sum_k \sum_{(v_i,v_j)\in E|v_i,v_j\in Q} x_{k,i,j} \leq |Q| - 1 \text{ where } \forall Q \subsetneq V$$

For the Sub-cycle removal constraint, it is required for every proper subset of V, leading to a situation where the number of constraints is $O(2^n)$, with n is the number of vertices. This situation may be impractical. The *Lazy Constraint* scheme [10] provides a solution to this difficulty. In the lazy constraint scheme, all constraints are initially removed. Then, whenever an *optimum* solution is obtained, the solution's proper feasibility is checked. If the solution has no sub-cycle, it must be exactly optimum. Otherwise, the corresponding sub-cycle removal constraint is added, and the new problem is solved. This scheme may add $O(2^n)$ constraints at worst. However, in practical cases, the number of added constraints can be much smaller.

B. *Reachable vertex set*

Using the formulation mentioned in the previous section, the two-step optimization is realized in this section. Initially, *Naive Method* is presented in Algorithm 1.

Algorithm 1 Naive Method

```
step 1: Minimize the maximum length of $z_k$, and the optimum length is $z_{\max}$.
step 2: Add the constraint $z_k \le z_{\max}$ for each $z_k$ and maximize the minimum length of $z_k$.
```

If each route r_k has an upper bound on its length, certain vertices may become inaccessible to r_k. To assess this impossibility, the concept of a *reachable vertex set* is proposed. The calculation method for the *lower bound of route* is defined in Algorithm 2 and Fig. 3 illustrates an example of the lower bound of route r_k.

Algorithm 2 Calculation of lower bound of route

```
Require: Graph $G = (V, E)$, source-pin set $S$, sink-pin set $T$
Ensure: Lower bound $L_{r_k}(v_i)$ of the route $r_k$ for each vertex $v_i$
    step 1: Calculate the length $\alpha_{s_k}(v_i)$ between $s_k$ and $v_i$ by breadth-first search.
    step 2: Calculate the length $\beta_T(v_i)$ between $v_i$ and an arbitrary sink-pin by breadth-first search.
    step 3: Calculate the lower bound or route $r_k$ $L_{r_k}(v_i) = \alpha_{s_k}(v_i) + \beta_T(v_i)$.
```

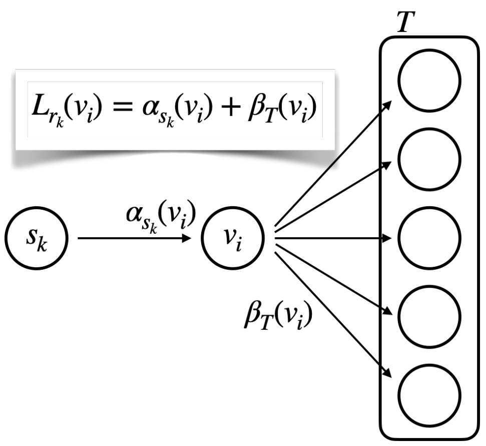

Fig. 3. Lower bound of route r_k : $L_{r_k}(v_i)$

The time complexity of the calculation of $L_{s_k}(v_i)$ consumes $O(mn)$ where the number of vertices and source-pins are n and m, respectively. Using the lower bound of route r_k, the definition of reachable vertex set is shown in definition 1.

Definition 1. Reachable vertex set For a given integer B, a set of vertices whose lower bound of route r_k is equal to or less than B is called *B-reachable vertex set* of r_k.

To enhance the speed of the Naive method (Algorithm 1), the reachable vertex set is employed. In the ILP formulation, if the vertex v_i is not in B-reachable vertex set for the route r_k, then $x_{k,i,j} = 0$ is set for all edges $(v_i, v_j) \in E$. Two modification points in Algorithm 1 are that 1. the minimization of the maximum length (step 1), and 2. the maximization of the minimum length (step 2).

For step 1, the upper bound of the maximum length $B_{\max}$ is required. The total-length minimization can be accomplished in polynomial time. To obtain a feasible solution, the total-length minimization is executed. The maximum length of its solution is then utilized as $B_{\max}$, and the minimization of the maximum length of r_k is solved. This step is referred to *Total-max initial (Ti)* step.

Next, for step 2, named *Limited min-max (Lmm)* step, the length of each route must be less than or equal to $z_{\max}$, which is the resulting length from step 1. Thus, for each route r_k, only the consideration of $z_{\max}$-reachable vertex set is sufficient.

As a result, *Ti_and_Lmm Method* is defined, where step 1, step 3, and step 4 correspond to the calculation of the reachable vertex set, the minimization of the maximum length, and the maximization of the minimum length, respectively.

Algorithm 3 Ti_and_Lmm (T-L) Method

```
step 1: Calculate $L_{r_k}(v_i)$ of the route $r_k$ for each vertex $v_i$ with Algorithm 2.
step 2: Solve the total-length minimization and the maximum length of the solution is set to $B_{\max}$.
step 3: (Ti-step) Minimize the maximum length of $z_k$ with $B_{\max}$-reachable vertex set, and the optimum length is $z_{\max}$.
step 4: (Lmm-step) Add the constraint $z_k \le z_{\max}$ for each $z_k$ and maximize the minimum length of $z_k$ with $z_{\max}$-reachable vertex set.
```

From the process log of Algorithm 3, the following observation is obtained. (a) The *optimum* solution can be obtained in a very early stage. (b) However, confirming its *optimality* requires a long optimization process. Thus, an iterative method is proposed, which checks whether a better solution exists than the best so far, as shown in Algorithm 4.

In this algorithm, step 3.1 considers the constraint

```
Algorithm 4 Iterative Decision (ID) Method
step 1: Calculate $L_{r_k}(v_i)$ of the route $r_k$ for each vertex $v_i$ with Algorithm 2.
step 2: Solve the total-length minimization and the maximum length of the solution is set to $B_{\max}$.
step 3: The Following steps are executed iteratively.
    step 3.1: Add the constraint $z_k \leq z_{\max} - 1$ for each $z_k$ and check whether a feasible solution exists or not with $(z_{\max} - 1)$-reachable vertex set.
    step 3.2: If exsits, reset $z_{\max}$ to the resultant maximum length and return step 3.1; Otherwise, escape this loop.
step 4: Add the constraint $z_k \leq z_{\max}$ for each $z_k$ and maximize the minimum length of $z_k$ with $z_{\max}$-reachable vertex set.
```

$z_k \leq z_{\max} - 1$ and checks the existence of feasible solution with $(z_{\max} - 1)$-reachable vertex set. Thus, in each step, fast convergence may be obtained because the domain becomes smaller than that of the previous steps.

IV. Experiments

The empirical validation of the proposed methods in section A will be conducted. The experimental environment is specified as follows: **CPU** Apple M2 Pro; **Memory** 32GB; **OS** macOS 13.2.1; **Language** Python 3.9.6; **Solver** Gurobi 10.0.1; **Threads** 8. The benchmark data used is the same as [4]. The specifications of the benchmark data are provided in Table I.

TABLE I
Benchmark data

Name	# Vertices	# Sources
E1	289	16
E2	239	16
B1	290	6
B2	683	12
B3	683	12
S1	90	6
S2	340	8
S3	800	12
F1	609	12

Firstly, data in Table I with fewer than 500 vertices are selected: E1, E2, B1, S1, and S2. In the experiments, the limitation of one ILP processing time is set to 10 minutes.

The run-time for E1, E2, B1, S1, and S2 are presented in the corresponding columns in Table II; Raw Naive, Lmm, Ti, T-L, and ID correspond to the results of Algorithm 1, only Lmm-step in Algorithm 3, only Ti-step in Algorithm 3, Algorithm 3, and Algorithm 4, respectively. Furthermore, Raw [4] and [8] show the results of the corresponding papers, respectively. Since [7] is an ILP-based optimization, it might be included into the comparison. But, [8] reported the optimization performance of [8] surpasses that of [7]. Thus, [7] is omitted from the comparison. The experimental environments are as follows: 1) [4]: Intel Core i7 4790K CPU, 32 GB memory, and Ubuntu 16.04 OS; 2) [8]: Intel Core i5-8500 CPUm 8 GB memory, CentOS 7(2007) OS, and minisat2.2.0 for the SAT solver. Each result indicates the resultant time in seconds, where underlined corresponds to the time-over.

From these results, it can be observed that the Naive method is much slower than the others, emphasizing the efficiency of using the reachable vertex set. In particular, T-L (Algorithm 3) method and ID (Algorithm 4) method demonstrate fast convergence. This observation suggests that restricting variables using the reachable vertex set is widely beneficial.

Next, the maximum length and the difference between the maximum and minimum length of the proposed method (Proposed), [4], and [8] are shown in Table III, where Max and Δ columns correspond to the maximum length and the difference maximum and minimum length, respectively.

In the table, the results of S1 and S2 from [8] are omitted since they cannot obtain results due to the memory limitation. Compared with the previous works, the proposed method performs better than [4] and comparably to [8]. Although [4] only needs less than 0.01 seconds for the optimization time, there is no possibility of refinement even with extended processing time since the method is deterministic. On the other hand, the proposed method's length performance of is comparable to [8]. [8] reported that the lack of memory prevented the output of S1 and S2. However, the method of [8] essentially require the exponential number of constraints, so a slightly larger memory size can not fundamentally enhance optimization ability.

Then, the results are shown for larger data in Table I, where they have 500 or more vertices. In this experiment, only T-L and ID are employed. Table IV shows the results of run-time in seconds. Compared with T-L and ID, ID tends to be faster than T-L. This result supports the observation that confirming optimality may be slow convergence. Unfortunately, the optimum solution of B3 and S3 cannot be obtained due to the long run-time. Furthermore, speed-up is needed.

Next, Table V shows the results of the maximum length and the difference between the maximum and minimum length of the proposed method and [4]. These results also demonstrate better solutions than those from [4].

Finally, the input data of S3 and the output by ID are shown in Fig. 4 and Fig. 5, respectively, where the source-pin and sink-pin correspond to red-colored and blue-colored circles, respectively, and the route is shown

TABLE II
Run-time [sec.]

Method	E1		E2		B1		S1		S2	
	Total		Total		Total		Total		Total	
	MaxMin	MinMax	MaxMin	MinMax	MaxMin	MinMax	MaxMin	MinMax	MaxMin	MinMax
Naive	78.58		605.45		2.23		10.93		53.15	
	1.83	76.75	5.38	600.08	0.45	1.77	0.30	10.63	18.81	34.34
Lmm	4.83		22.12		0.53		1.65		21.78	
	0.39	4.44	5.47	16.65	0.39	0.15	0.25	1.40	18.83	2.95
Ti	2.56		9.95		0.31		2.59		3.63	
	0.16	2.39	1.06	8.89	0.16	0.14	0.16	2.43	2.36	1.27
T-L	2.55		9.96		0.31		1.46		5.24	
	0.16	2.39	1.05	8.91	0.16	0.15	0.18	1.29	2.34	2.89
ID	2.50		9.53		0.33		1.23		3.86	
	0.18	2.32	0.61	8.92	0.18	0.14	0.15	1.08	0.98	2.88
[4]	0.003		0.003		0.001		0.000		0.002	
[8]	27.6		3.4		281.5		–		–	

TABLE III
Result of Maximum length (Max) and difference between maximum and minimum length (Δ)

Name	Proposed		[4]		[8]	
	Max	Δ	Max	Δ	Max	Δ
E1	10	0	10	0	10	0
E2	14	6	14	6	14	6
B1	17	1	17	1	17	1
S1	12	7	12	8	–	
S2	15	1	16	2	–	

TABLE IV
Run-time [sec.]

Name	T-L		ID	
	MaxMin	MinMax	MaxMin	MinMax
B2	19.83	7.43	2.19	6.10
B3	176.52	600.19	16.88	600.02
S3	612.23	600.11	643.95	110.21
F1	4.37	5.04	0.92	5.01

TABLE V
Result of Maximum length (Max) and difference between maximum and minimum length (Δ)

Name	Proposed		[4]	
	Max	Δ	Max	Δ
B2	30	1	30	1
B3	40	8	46	8
S3	28	1	30	1
F1	15	0	19	4

by the red bold line.

In conclusion, the proposed method demonstrates the capability to output the optimum solution or the best-known solution, making it significantly more efficient than previous works.

V. Conclusion Remarks

This paper proposed a fast ILP method for the set-pair routing problem, addressing limitations in previous works categorized as fast heuristics without guarantees of obtaining the optimum solution or exact methods with slow convergence within practical timeframes. The proposed method presented an exact approach capable of convergence within practical time limits, and its efficiency is empirically confirmed.

As part of future work, further speed-up strategies are needed. The size of the model generally impacts the convergence speed of ILP optimization, emphasizing the importance of detecting and eliminating unnecessary variables. Additionally, exploring various speed-up methods for ILP-based formulations presents opportunities for future research.

Acknowledge

Special thanks to Prof. Shimpei Sato from Shinshu University for providing the benchmark data.

References

[1] Y. Kohira, S. Suehiro, and A. Takahashi. A fast longer path algorithm for routing grid with obstacles using biconnectivity based length upper bound. *IEICE Trans. Fundamentals* vol. E92-A, No. 12 (2009), pp.2971–2978.

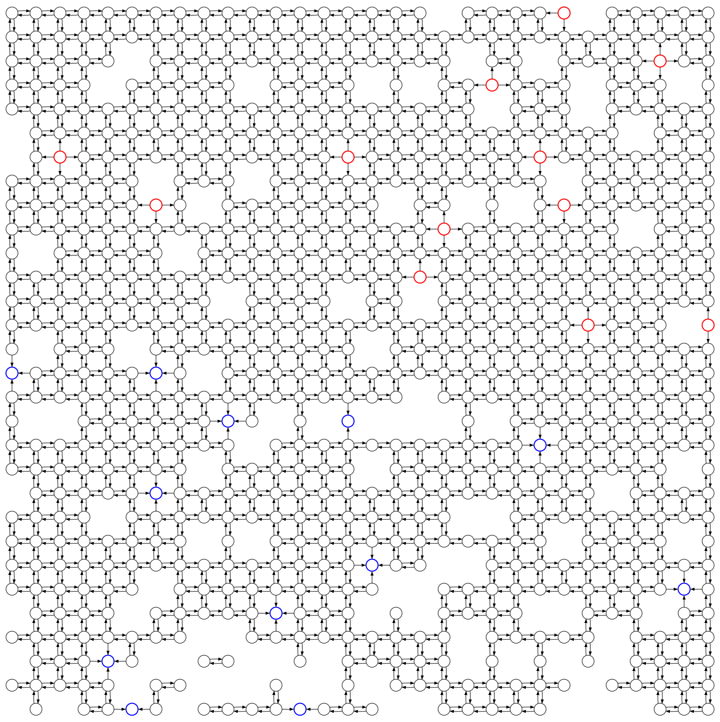
Fig. 4. Input data of S3

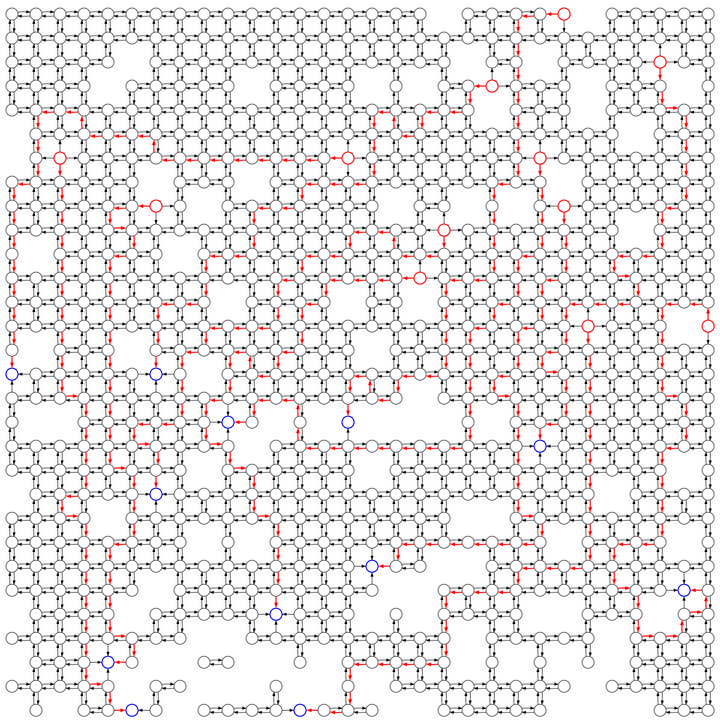
Fig. 5. Output from ID

[2] Y. Kohira and A. Takahashi. CAFE router: A fast connectivity aware multiple nets routing algorithm for routing grid with obstacles. *IEICE Trans. Fundamentals* vol. E93-A, No. 12 (2010), pp.2380–2388.

[3] A. Takahashi. On set pair routing problem. *IEICE Technical Report* vld2011, 44 (Sept. 2011), pp.23–28. in Japanese.

[4] S. Sato, K. Akagi, and A. Takahashi. A fast length matching routing pattern generation method for set-pair routing problem using selective pin-pair connections. *IEICE Trans. Fundamentals* Vol. E103 – A, No. 9 (2020), pp.1037–1044.

[5] Y. Nakatani and A. Takahashi. A length matching routing algorithm for set-pair routing problem. *IEICE Trans. Fundamentals* Vol. E98-A, No. 12 (2015), pp.2565–2571.

[6] K. Akagi, S. Sato, and A. Takahashi. Target pin-pair selection algorithm using minimum maximum-edge-weight matching for set-pair routing. *in Proceeding of SASIMI 2018* (2018), pp.337–342.

[7] S. Hara and K. Fujiyoshi. Max Length and Length Difference Minimization for Set Pair Routing Problem with ILP. *IEICE Technical Report* vld2017, 60 (Nov. 2017), pp.337–342. in Japanese.

[8] K. Nagakura, R. Yokoya, and K. Fujiyoshi. A Routing Method by SAT for Set-Pair Routing Problem. *IEICE Technical Report* vld2022, 21 (Nov. 2022), pp.13–18. in Japanese.

[9] A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks* Vol. 12, No. 3 (1982), pp.277–286.

[10] Gurobi. tsp.py. `https://www.gurobi.com/documentation/10.0/examples/tsp_py.html`.