TUGAS AKHIR

Smart Travel Route Planner: Rute Liburan Optimal dengan Algoritma Tabu Search (Minimasi Biaya & Waktu)

Oleh:

Andre Kurniawan 2125250006

Program Studi

Informatika

Fakultas Ilmu Komputer dan Rekayasa

Universitas Multi Data Palembang

2025

1. LATAR BELAKANG

Perencanaan perjalanan wisata sering kali menjadi tantangan tersendiri bagi para wisatawan, terutama ketika harus menentukan rute terbaik yang mencakup berbagai destinasi dalam waktu dan anggaran terbatas. Banyaknya pilihan lokasi wisata, keterbatasan waktu, serta biaya transportasi dan akomodasi yang beragam membuat proses perencanaan menjadi kompleks. Dalam konteks ini, perencanaan rute perjalanan yang efisien tidak hanya penting untuk kenyamanan, tetapi juga dapat meningkatkan pengalaman wisata secara keseluruhan.

Permasalahan ini menyerupai *Traveling Salesman Problem* (TSP), yaitu bagaimana menemukan rute terpendek yang mengunjungi sejumlah titik dan kembali ke titik awal, namun dalam konteks wisata, faktor biaya dan waktu menjadi variabel yang sangat menentukan. Oleh karena itu, dibutuhkan pendekatan komputasional cerdas yang mampu mengoptimalkan perencanaan rute berdasarkan dua parameter utama: minimasi biaya dan minimasi waktu tempuh.

Melalui penggabungan data lokasi wisata, estimasi waktu tempuh, serta biaya perjalanan, sistem ***Smart Travel Route Planner*** berbasis algoritma *Tabu Search* diharapkan mampu menghasilkan rekomendasi rute terbaik yang tidak hanya efisien dari segi waktu dan biaya, tetapi juga fleksibel terhadap preferensi pengguna. Dengan demikian, aplikasi ini dapat menjadi solusi praktis dan cerdas dalam membantu wisatawan merencanakan liburan yang optimal.

Tabu Search merupakan algoritma metaheuristik yang menggunakan pendekatan iteratif dan memori jangka pendek (*tabu list*) untuk menghindari siklus dan menjelajahi ruang solusi yang lebih luas. Dengan pendekatan ini, sistem dapat

menemukan rute wisata yang lebih optimal dalam hal biaya dan waktu. Oleh karena itu, proyek ini bertujuan mengembangkan aplikasi *Smart Travel Route Planner* dengan memanfaatkan algoritma Tabu Search untuk membantu wisatawan merencanakan perjalanan yang optimal.

2. TUJUAN

Tujuan dari projek ini adalah untuk merancang dan mengaplikasikan rute liburan kegiatan yang berbasis preferensi pengguna untuk menemukan Solusi yang optimal. Secara lebih rinci, tujuan dari proyek ini adalah sebagai berikut:

1. Bagaimana merancang sistem perencanaan rute perjalanan wisata yang mampu meminimalkan biaya dan waktu?
2. Bagaimana algoritma Tabu Search dapat diterapkan dalam pencarian rute perjalanan wisata?
3. Sejauh mana efektivitas algoritma Tabu Search dibandingkan metode konvensional seperti algoritma Greedy?

3. Studi Literatur / Tinjauan Pustaka

3.1 Traveling Salesman Problem (TSP)

Traveling Salesman Problem (TSP) adalah permasalahan optimasi klasik dalam ilmu komputer dan matematika yang bertujuan menemukan rute terpendek yang mengunjungi sejumlah kota (titik) tepat satu kali dan kembali ke titik awal. Dalam konteks aplikasi wisata, TSP sangat relevan karena

permasalahan ini dapat dimodelkan sebagai upaya mengunjungi beberapa destinasi wisata dengan biaya dan waktu seminimal mungkin.

TSP termasuk dalam kategori NP-Hard, artinya tidak ada algoritma eksak yang mampu menyelesaikan persoalan ini secara efisien untuk ukuran input yang besar.

3.2 Algoritma Greedy

Algortima Greedy adalah sebuah metode pencarian Solusi yang menghasilkan keputusan secara lokal pada setiap Langkah dengan harapan menghasilkan yang optimal. Dalam perencanaan rute, Greedy biasanya memilih destinasi yang termurah secara berurutan.

3.3 Algoritma Tabu Search

Tabu search merupakan algoritma metaheuristic yang dikembangkan oleh (Fred Glover et al., n.d.)pada tahun 1896. Berbeda dengan algoritma pencarian lokal biasa, Tabu Search menggunakan memori jangka pendek yang disebut dengan tabu list untuk menyimpan Solusi sebelumnya dan menghindari perulangan terhadap Solusi yang sama.

4. Metodelogi

A. Data

Data yang digunakan dalam penelitian ini terbagi menjadi dua kategori utama, yaitu data masukan secara real-time melalui antarmuka aplikasi (GUI) dan data uji (testing data) yang digunakan untuk pengujian kinerja algoritma.

B. Algoritma

Pada projek ini menggunakan algoritma tabu search untuk menemukan Solusi yang optimal dalam menentukan rute perjalanan wisata. Tabu Search dipilih karena kemampuannya untuk menghindari Solusi yang tidak optimal dan menjelajahi secara menyeluruh dari pda algoritma sederhana seperti greedy

```
class TabuSearchPlanner:
    def __init__(self, master):
        self.master = master
        self.master.title("Smart Travel Route Planner")
        self.locations = []
        self.weights = []
        self.penalty_weight = tk.DoubleVar(value=0.5)
        self.coords = []  # Untuk menyimpan koordinat visualisasi
        self.setup_ui()
        self.tabu_process = None

        # Variabel untuk tracking optimum solution
        self.no_improvement_count = 0
        self.max_no_improvement = 15  # Berhenti jika tidak ada peningkatan setelah 15 iterasi
```

Algoritma tabu search

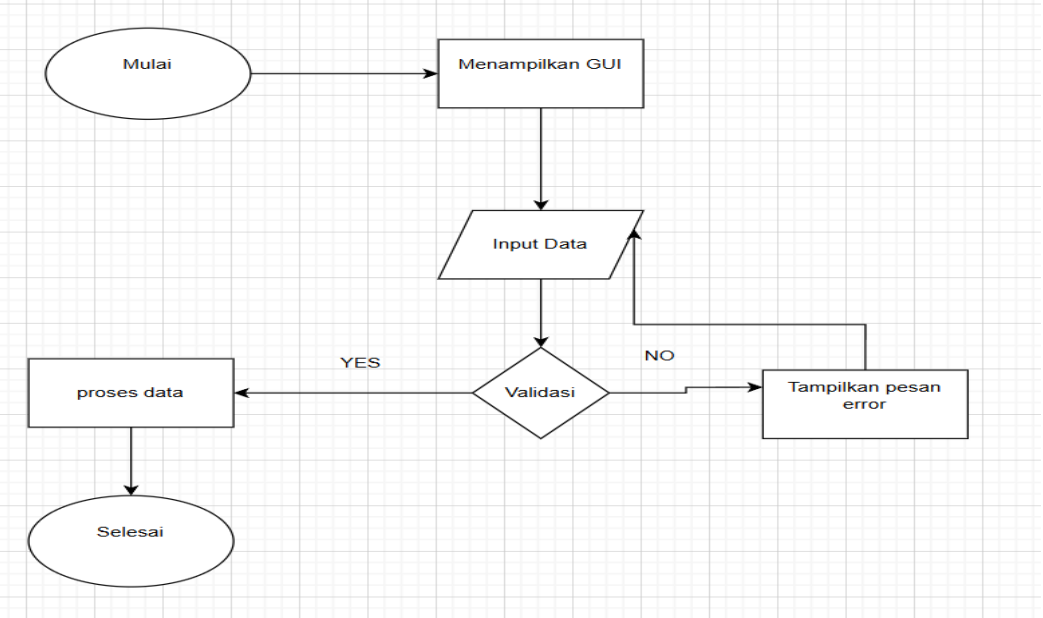
C. Rancangan program

1. Arsitektur program

Aplikasi rute liburan optimal dengan algoritma tabu search. Aplikasi ini menunjukan dimana Langkah demi Langkah proses kerja tabu search. GUI tersebut dibuat dengan menggunak Thinker yaitu salah satu *library* yang ada di *python.*

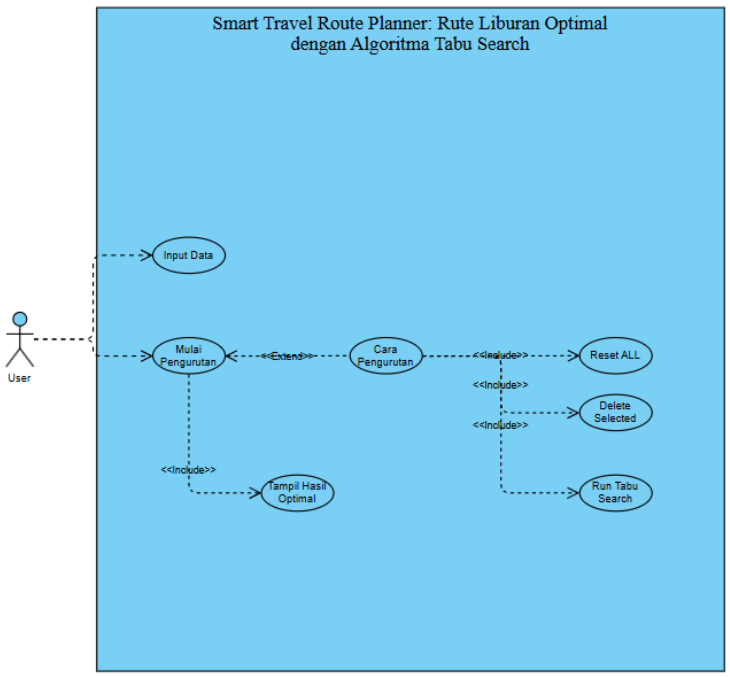
2. Flowchart

Flowchart menjelaskan alur program untuk aplikasi rure liburan menggunakan algoritma tabu search . Flowchart ini dapat membantu dalam memahami cara kerja program dan apa saja yang akan ditampilkan kepada pengguna, berikut tampilan flowchart nya



3. Diagram use Case

Diagram *use case* dalam aplikasi penyortiran ini digunakan untuk menggambarkan interaksi antara pengguna dengan sistem serta dapat membantu pengguna dengan visualisasi fitur-fitur yang tersedia dan yang dapat digunakan. Berikut gambar diagram usecase



UseCase Diagram

4. Kode program

```
import tkinter as tk
from tkinter import ttk, messagebox
import random
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

class TabuSearchPlanner:
    def __init__(self, master):
        self.master = master
        self.master.title("Smart Travel Route Planner")
        self.locations = []
        self.weights = []
        self.penalty_weight = tk.DoubleVar(value=0.5)
        self.coords = []  # Untuk menyimpan koordinat visualisasi
        self.setup_ui()
        self.tabu_process = None

        # Variabel untuk tracking optimum solution
        self.no_improvement_count = 0
        self.max_no_improvement = 15  # Berhenti jika tidak ada peningkatan setelah 15 iterasi
```

```
def setup_ui(self):
    main_frame = tk.Frame(self.master)
    main_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    # Input Frame
    input_frame = tk.LabelFrame(main_frame, text="Input Lokasi", padx=5, pady=5)
    input_frame.pack(fill=tk.X, pady=5)

    tk.Label(input_frame, text="Nama Lokasi:").grid(row=0, column=0, padx=5)
    self.entry_name = tk.Entry(input_frame, width=20)
    self.entry_name.grid(row=0, column=1, padx=5)

    tk.Label(input_frame, text="Bobot Biaya:").grid(row=0, column=2, padx=5)
    self.entry_weight = tk.Entry(input_frame, width=10)
    self.entry_weight.insert(0, "1.0")
    self.entry_weight.grid(row=0, column=3, padx=5)

    tk.Button(input_frame, text="Add", command=self.add_location, width=8).grid(row=0, column=4, padx=5)

    tk.Label(input_frame, text="Denda per Unit (>300km):").grid(row=1, column=0, padx=5)
    tk.Entry(input_frame, textvariable=self.penalty_weight, width=10).grid(row=1, column=1, padx=5, sticky='w')

    # Locations Table Frame
    table_frame = tk.LabelFrame(main_frame, text="Daftar Lokasi", padx=5, pady=5)
    table_frame.pack(fill=tk.BOTH, expand=True, pady=5)

    columns = ('no', 'lokasi', 'bobot', 'biaya_per_langkah', 'total_biaya')
    self.tree = ttk.Treeview(table_frame, columns=columns, show='headings')
```

```
# Configure columns
self.tree.heading('no', text='No')
self.tree.heading('lokasi', text='Lokasi')
self.tree.heading('bobot', text='Bobot')
self.tree.heading('biaya_per_langkah', text='Biaya per Langkah')
self.tree.heading('total_biaya', text='Total Biaya')

self.tree.column('no', width=50, anchor='center')
self.tree.column('lokasi', width=150)
self.tree.column('bobot', width=80, anchor='center')
self.tree.column('biaya_per_langkah', width=120, anchor='center')
self.tree.column('total_biaya', width=100, anchor='center')

scrollbar = ttk.Scrollbar(table_frame, orient="vertical", command=self.tree.yview)
self.tree.configure(yscrollcommand=scrollbar.set)

self.tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

# Button Frame
button_frame = tk.Frame(main_frame)
button_frame.pack(fill=tk.X, pady=5)

tk.Button(button_frame, text="Delete Selected", command=self.delete_selected, width=15).pack(side=tk.LEFT, padx=5)
tk.Button(button_frame, text="Run Tabu Search", command=self.run_tabu_search, width=15).pack(side=tk.LEFT, padx=5)
tk.Button(button_frame, text="Reset All", command=self.reset_all, width=15).pack(side=tk.RIGHT, padx=5)

# Best Solution Frame - NEW
best_solution_frame = tk.LabelFrame(main_frame, text="Solusi Terbaik Saat Ini", padx=5, pady=5)
best_solution_frame.pack(fill=tk.X, pady=5)

self.best_solution_text = tk.Text(best_solution_frame, height=3, width=80, wrap=tk.WORD)
self.best_solution_text.pack(fill=tk.X, padx=2, pady=2)
self.best_solution_text.config(state=tk.DISABLED, bg='#f0f0f0')

# Process Table Frame
process_frame = tk.LabelFrame(main_frame, text="Proses Tabu Search", padx=5, pady=5)
```

```python
def update_best_solution_display(self):
    """Update the best solution display with current best solution"""
    if self.best_solution is None:
        return

    self.best_solution_text.config(state=tk.NORMAL)
    self.best_solution_text.delete(1.0, tk.END)

    named_route = " → ".join([self.locations[i] for i in self.best_solution])
    best_cost, best_distance, best_penalty = self.calculate_route_cost(self.best_solution)

    best_solution_info = (f"Rute: {named_route}\n"
                          f"Total Jarak: {best_distance:.2f} km | "
                          f"Total Denda: {best_penalty:.2f} | "
                          f"Total Biaya: {best_cost:.2f}")

    self.best_solution_text.insert(tk.END, best_solution_info)
    self.best_solution_text.config(state=tk.DISABLED)

    # Highlight the text box to make it more visible when updated
    self.best_solution_text.config(bg='#e6ffe6')  # Light green background
    self.master.after(500, lambda: self.best_solution_text.config(bg='#f0f0f0'))  # Reset after 500ms
```

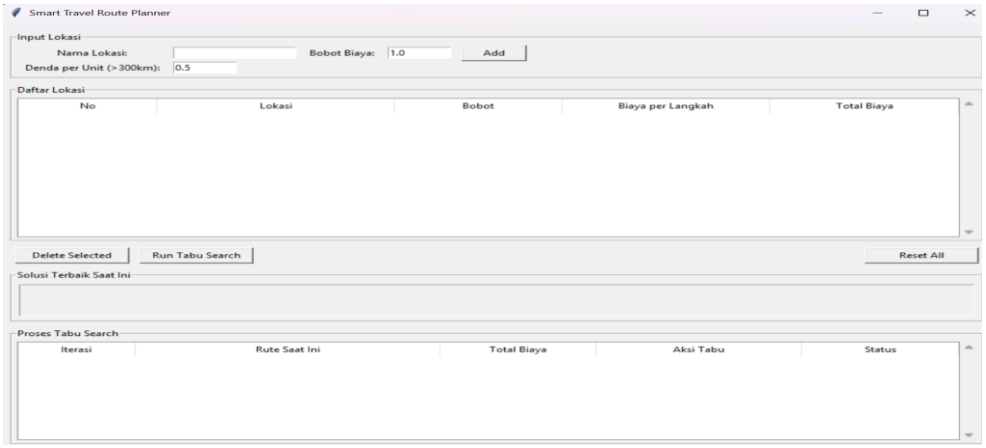
5. IMPLEMENTASI

1. INPUT DAN OUTPUT PROGRAM

Pada pengujian ini, pengguna akan menginput nama Lokasi dan bobot secara manual dari *text field* yang sudah dibuat pada aplikasi tersebut, setelah pengguna memasukan nama Lokasi dan bobot akan di tampilkan nama Lokasi dan bobot pada tabel, kemudian program akan mencarikan Solusi yang optimal.

2. Tampilan Program

Graphical User Interface(GUI) pada program ini dibuat menggunakan library python Thinker yang memberikan untuk menampilkan UI yang interaktif. GUI ini akan menampilkan textfield berupa nama Lokasi, bobot dan total denda, daftar Lokasi, tombol *delete selected, reset all, run tabu search*, proses tabu search dan Solusi terbaik saat ini. Berikut gambar di bawah ini



3. Penjelasan cara penggunaan

pada aplikasi *Smart Travel Route Planner*: Rute Liburan Optimal dengan Algoritma Tabu Search untuk membantu pengguna dalam mengatur rute berdasarkan bobot. Berikut ini adalah Langkah-langkah penggunaan aplikasi yang dapat diikuti oleh pengguna:

Langkah-langkah Pengguna

1. **Persiapan awal**

   a. Pastikan *python* yang ada di komputer sudah terinsatall aplikasi yang Bernama *Python*(disarankan menggunakan versi terbaru).

   b. Unduh seluruh data program dari link berikut (https://github.com/Deadring/SA.git) yang berupa tabu.py

   c. Cara menjalankan program

      1. Buka terminal pada *visual studio code* atau *Command Prompt* yang tersedia pada komputer masing-masing

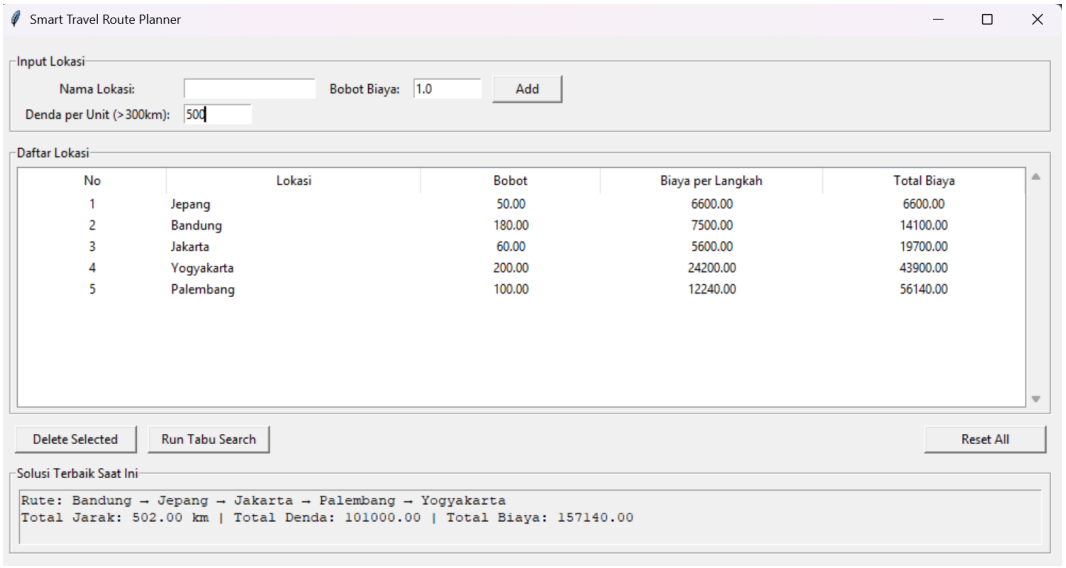
      2. Kemudian buka file yang tersimpan di direktori yang disimpan

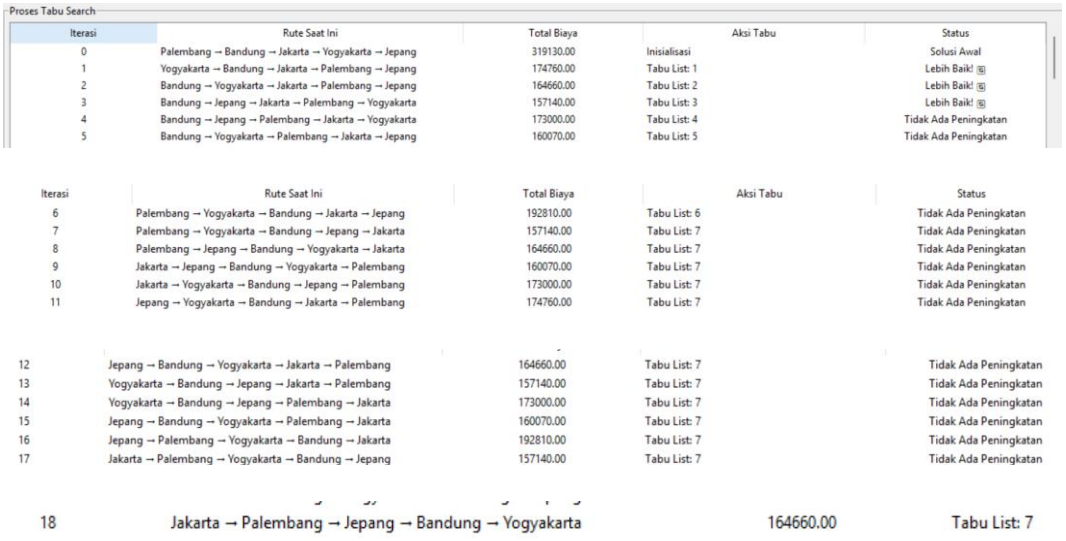
3. Ketika sudah membuka file yang tersimpan jalankan perintah di terminal dengan nama *pyhton* tabu.py

6. Pengujian

pada pengujian tersebut dilakukan untuk memastikan bahwa aplikasi Smart Travel Route Planner menggunakan algoritma tabu search berjalan dengan baik serta dirancang dengan baik

berikut hasil pengujian nya:





7. Kesimpulan dan saran

7.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi Smart Travel Route Planner menggunakan algoritma tabu search, dapat disimpulkan sebagai berikut:

1. Sistem berhasil menghasilkan rute perjalanan wisata yang optimal berdasarkan dua parameter utama, yaitu minimasi biaya dan waktu tempuh.
2. Algoritma Tabu Search terbukti efektif dalam mencari solusi mendekati optimal, dengan kemampuan menghindari jebakan *local optimum* yang sering terjadi pada pendekatan heuristik sederhana seperti Greedy.

7.2 Saran

Untuk program tersebut , beberapa saran berikut:

- Integrasi Data Real-time: Sistem dapat ditingkatkan dengan mengintegrasikan data real-time seperti kondisi lalu lintas, cuaca, dan harga transportasi dinamis untuk meningkatkan akurasi estimasi.
- Personalisasi Lebih Lanjut: Sistem dapat dikembangkan untuk mempertimbangkan preferensi spesifik pengguna seperti minat wisata (kuliner, budaya, alam) dalam penentuan rute.

- Dukungan Multi-Hari: Menambahkan fitur perencanaan perjalanan multi-hari agar sistem dapat mengatur jadwal menginap dan membagi destinasi ke dalam beberapa hari secara efisien.

Daftar Pustaka

Ariantini, M. S., & Dirgayusari, A. M. (2021). Implementasi Metode Tabu Search Dalam Penjadwalan Menggunakan Analisa Pieces. *INFORMAL: Informatics Journal*, *6*(2), 62. https://doi.org/10.19184/isj.v6i2.23811

Barbarosoglu, G., & Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research*, *26*(3), 255–270. https://doi.org/10.1016/S0305-0548(98)00047-1

Fred Glover, M. Laguna, & Laguna, M. (n.d.). *Tabu Search /// Tabu search*. 8–32.

韩丽敏, 韦有双冯允成. (1998). ***关于****Tabu Search****算法收****敛性的研究***. *3*, 621–757.