

INDIVIDUAL WORK: Deadline: 22/02/2023

Sent the pdf file with answers by email to

leandro.coelho@ufpr.br **SUBJECT:** Computational work 03

Computational work 03

Combinatorial optimization

Leandro dos Santos Coelho

Pontifícia Universidade Católica do Paraná (PUCPR), Escola Politécnica

Pós-Graduação em Engenharia de Produção e Sistemas (PPGEPS), Graduação em Engenharia de Controle e Automação (Mecatrônica)

Rua Imaculada Conceição, 1155, CEP 80215-901, Curitiba, PR, Brasil

Universidade Federal do Paraná (UFPR), Graduação e Pós-Graduação em Engenharia Elétrica, Campus Centro Politécnico

Av. Cel. Francisco H. dos Santos, 100, CEP 81530-000, Curitiba, PR, Brasil

e-mail: leandro.coelho@pucpr.br; lscoelho2009@gmail.com; leandro.coelho@ufpr.br

Currículo Lattes: <http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4792095Y4>

Google Scholar: <https://scholar.google.com/citations?user=0X7VkC4AAAAJ&hl=pt-PT>

Computational work

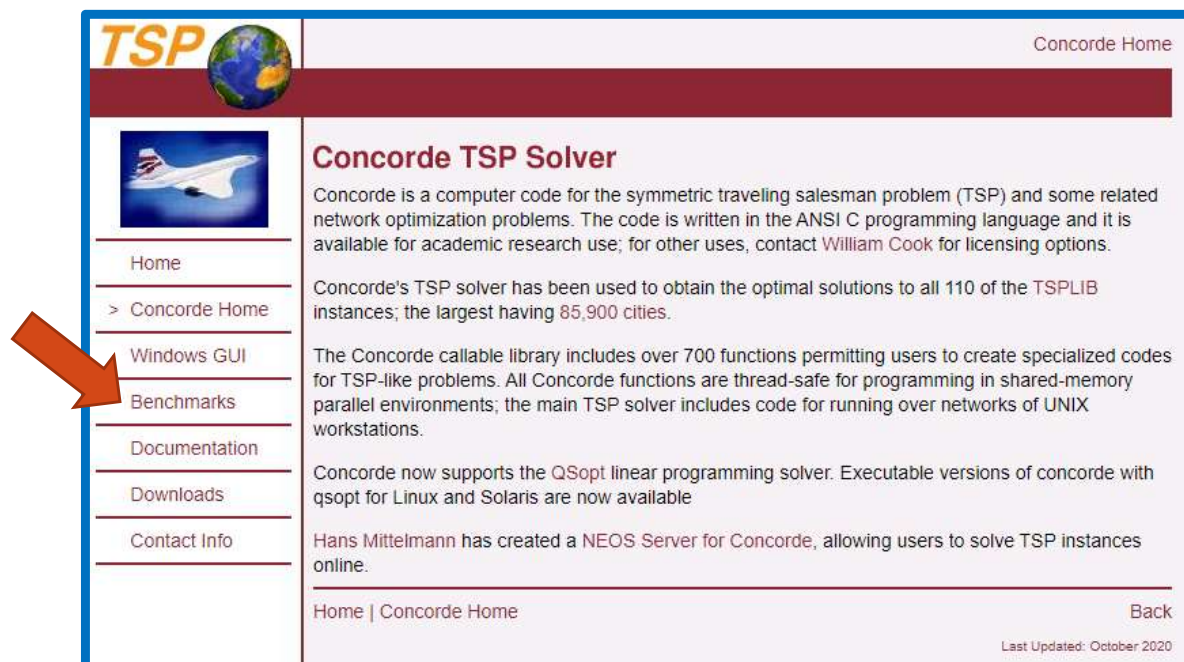
Software: Matlab, R, Python, Julia, Excel
Zipped report and source codes
Use IEEE Access template

Compare the performance of **3 optimization** approaches (classical approaches and/or heuristics and/or metaheuristics) to the **4 cases** (minimum with 10 cities) of **traveling salesman problem (TSP)**.

In the case of metaheuristics utilization includes a table with (**min, max, mean, median and standard deviation**) objective functions values (**10** runs with different initial solution for each run) to all cases must be presented and discussed.

Example of cases 1:

TSP	Number Of Cities
city10	10
city15	15
city29	29
city30	30
city35	35
att48	48
eil51	51
city51	51
eil75	75



Comments

>> rng(run_integer_number, 'twister') % for reproducibility in 30 runs OR

>> rand('state', run_integer_number)

Build a TABLE with results for 10 runs related to objective function values (min, max, mean, std)

		Optimizer (setup)	Minimum f	Mean f	Median f	Maximum f	Standard deviation, f
case 1	{	Optimizer 1					
		Optimizer 2					
		Optimizer 3					
case 2	{	Optimizer 1					
		Optimizer 2					
		Optimizer 3					
case 3	{	Optimizer 1					
		Optimizer 2					
		Optimizer 3					
case 4	{	Optimizer 1					
		Optimizer 2					
		Optimizer 3					



Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems



Eneko Osaba^{a,b,*}, Xin-She Yang^b, Fernando Diaz^a, Pedro Lopez-Garcia^a, Roberto Carballedo^a

^a Deusto Institute of Technology (DeustoTech), University of Deusto, Av. Universidades 24, Bilbao 48007, Spain

^b School of Science and Technology, Middlesex University, Hendon Campus, London, NW4 4BT, United Kingdom

ARTICLE INFO

Article history:

Received 9 April 2015

Received in revised form

10 September 2015

Accepted 23 October 2015

Keywords:

Bat algorithm

Traveling Salesman Problem

Genetic algorithms

Combinatorial optimization

Routing problems

ABSTRACT

Bat algorithm is a population metaheuristic proposed in 2010 which is based on the echolocation or bio-sonar characteristics of microbats. Since its first implementation, the bat algorithm has been used in a wide range of fields. In this paper, we present a discrete version of the bat algorithm to solve the well-known symmetric and asymmetric Traveling Salesman Problems. In addition, we propose an improvement in the basic structure of the classic bat algorithm. To prove that our proposal is a promising approximation method, we have compared its performance in 37 instances with the results obtained by five different techniques: evolutionary simulated annealing, genetic algorithm, an island based distributed genetic algorithm, a discrete firefly algorithm and an imperialist competitive algorithm. In order to obtain fair and rigorous comparisons, we have conducted three different statistical tests along the paper: the Student's *t*-test, the Holm's test, and the Friedman test. We have also compared the convergence behavior shown by our proposal with the ones shown by the evolutionary simulated annealing, and the discrete firefly algorithm. The experimentation carried out in this study has shown that the presented improved bat algorithm outperforms significantly all the other alternatives in most of the cases.

© 2015 Elsevier Ltd. All rights reserved.

Example of cases 2

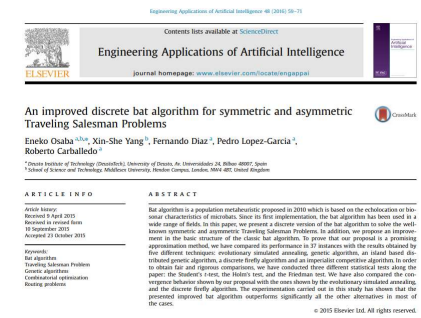
Parametrization of the IBA, BA1 and BA2 for the TSP and ATSP.

IBA		BA1		BA2	
Parameter	Value	Parameter	Value	Parameter	Value
Population size	50	Population size	50	Population size	50
Movement functions	2-opt & 3-opt	Movement function	2-opt	Movement function	3-opt
Initial A_i^0	Random number in [0,7,1,0]	Initial A_i^0	Random number in [0,7,1,0]	Initial A_i^0	Random number in [0,7,1,0]
Initial r_i^0	Random number in [0,0,0,4]	Initial r_i^0	Random number in [0,0,0,4]	Initial r_i^0	Random number in [0,0,0,4]
α & γ	0.98	α & γ	0.98	α & γ	0.98

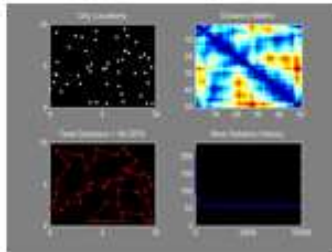
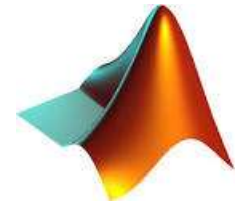
- Improved Bat Algorithm (IBA)
- Discrete Firefly Algorithm (DFA)
- Discrete Imperialist Competitive Algorithm (DICA)

Results of the proposed IBA, DFA and DICA for the TSP and ATSP.

Instance		IBA				DFA				DICA			
Name	Optima	Avg.	Best	S. dev.	Time	Avg.	Best	S. dev.	Time	Avg.	Best	S. dev.	Time
Oliver30	420	420.0	420	0.0	0.4	420.0	420	0.0	0.4	420.0	420	0.0	0.5
Eilon50	425	427.4	425	1.3	1.5	427.2	425	1.8	1.3	427.9	425	2.1	1.4
Eil51	426	428.1	426	1.6	1.7	430.8	426	2.3	1.6	432.3	426	3.1	1.8
Berlin52	7542	7542.0	7542	0.0	2.1	7542.0	7542	0.0	2.2	7542.0	7542	0.0	2.5
St70	675	679.1	675	2.8	3.9	685.3	675	4.0	4.3	684.7	675	3.7	4.1
Eilon75	535	547.4	535	3.9	4.5	543.6	535	5.3	4.6	551.7	537	6.8	5.6
Eil76	538	548.1	539	3.8	5.1	556.8	543	4.9	5.3	557.6	544	5.8	5.3
KroA100	21,282	21,445.3	21,282	116.5	10.6	21,483.6	21,282	163.7	10.3	21,500.3	21,282	183.4	10.8
KroB100	22,140	22,506.4	22,140	221.3	11.1	22,604.8	22,183	243.9	11.6	22,599.7	22,180	244.9	11.3
KroC100	20,749	21,050.0	20,749	164.7	12.0	21,096.3	20,756	148.3	12.8	21,103.9	20,756	161.1	11.7
KroD100	21,294	21,593.4	21,294	141.6	11.7	21,683.8	21,408	163.7	12.4	21,666.8	21,399	174.0	12.6
KroE100	22,068	22,349.6	22,068	169.6	11.4	22,413.0	22,079	183.0	11.6	22,453.3	22,083	196.9	11.7
Eil101	629	646.4	634	4.9	13.1	659.0	643	8.1	13.3	663.8	644	9.6	12.0
Pr107	44,303	44,793.8	44,303	232.4	12.1	44,790.4	44,303	227.3	12.6	44,803.3	44,303	302.7	12.9
Pr124	59,030	59,412.1	59,030	265.9	18.5	59,404.3	59,030	257.9	18.8	59,436.9	59,030	299.4	19.0
Pr136	96,772	99,351.2	97,547	707.2	23.4	99,683.7	97,716	831.3	24.1	99,583.7	97,736	848.9	24.0
Pr144	58,537	58,876.2	58,537	295.6	30.3	58,993.3	58,546	300.1	30.9	59,070.9	58,563	323.0	30.7
Pr152	73,682	74,676.9	73,921	426.5	31.0	74,934.3	74,033	483.7	32.1	74,886.7	74,052	513.9	32.0
Pr264	49,135	50,908.3	49,756	887.0	92.5	51,837.0	50,491	760.8	93.0	51,943.6	50,553	863.7	94.1
Pr299	48,191	49,674.1	48,310	1200.1	147.2	49,839.7	48,579	1305.4	149.1	49,880.3	48,600	1413.7	150.3
Pr439	107,217	115,256.4	111,538	3825.8	201.9	115,558.2	111,967	4009.1	202.4	115,763.1	111,983	4219.6	203.7
Pr1002	259,047	274,419.7	270,016	3617.8	681.7	277,344.7	272,003	4731.6	682.0	277,308.1	272,082	4293.7	684.6
br17	39	39.0	39	0.0	0.2	39.0	39	0.0	0.2	39.0	39	0.0	0.3
ftv33	1286	1318.1	1286	25.7	2.2	1320.9	1286	21.9	2.8	1324.6	1286	28.3	2.9
ftv35	1473	1493.7	1473	8.0	2.5	1498.8	1473	10.4	2.7	1490.6	1473	11.9	2.8
ftv38	1530	1562.0	1530	13.7	3.1	1560.4	1530	14.6	3.0	1565.6	1530	15.8	3.2
p43	5620	5620.0	5620	0.0	3.0	5620.0	5620	0.0	2.8	5620.0	5620	0.0	3.1
ftv44	1613	1683.7	1613	27.2	5.0	1690.8	1620	32.3	5.1	1694.3	1622	54.0	5.3
ftv47	1776	1863.6	1796	39.3	4.7	1858.3	1795	63.4	5.5	1873.0	1799	70.1	5.8
ry48p	14,422	14,544.8	14,422	79.7	4.2	14,694.4	14,453	94.7	4.4	14,689.8	14,463	73.6	5.4
ft53	6905	7294.1	7001	196.9	6.5	7302.0	6993	186.4	6.8	7320.1	7002	200.3	6.9
ftv55	1608	1737.5	1608	50.5	6.9	1790.6	1628	64.0	7.0	1801.4	1630	83.0	7.2
ftv64	1839	1999.2	1879	68.2	7.2	2041.6	1903	73.4	7.0	2040.8	1900	81.3	7.3
ftv70	1950	2233.2	2111	48.8	8.1	2290.8	2173	60.0	7.8	2322.6	2167	63.3	8.3
ft70	38,673	40,309.7	39,901	237.2	8.2	40,694.8	39,668	494.6	8.5	40,699.7	39,660	534.9	8.8
kro124p	36,230	39,213.7	37,538	947.5	15.4	41,637.5	39,438	1094.7	15.8	41,608.3	39,400	1116.8	15.7
rbg323	1326	1640.9	1615	30.4	243.6	1634.7	1599	34.6	245.1	1639.7	1600	31.1	247.0



Source codes



Traveling Salesman Problem - Genetic Algorithm

version 1.3 (9.62 KB) by Joseph Kirk

Finds a near-optimal solution to a TSP using a GA

<https://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm>

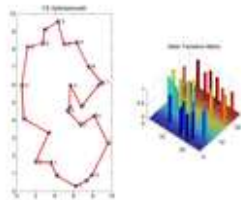


Dynamic Programming solution to the TSP

version 1.0 (3.04 KB) by Elad Kivelevitch

This function solves the Traveling Salesman Problem (TSP) using Dynamic programming (DP).

<https://www.mathworks.com/matlabcentral/fileexchange/31454-dynamic-programming-solution-to-the-tsp>

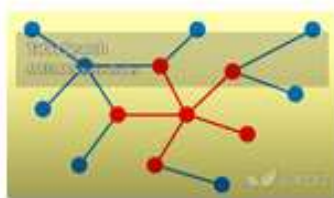
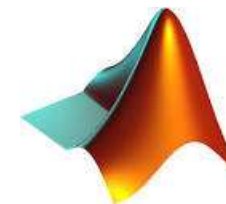


Cross Entropy TSP Solver

version 1.0 (68.7 KB) by Sebastien PARIS

Solve TSP problem with a CE method.

<https://www.mathworks.com/matlabcentral/fileexchange/4821-cross-entropy-tsp-solver>



Tabu Search (TS)

version 1.0 (16.5 KB) by Yarpiz

A structured implementation of Tabu Search (TS) in MATLAB for TSP and n-Queens Problem

<https://www.mathworks.com/matlabcentral/fileexchange/52902-tabu-search--ts->



TSPSEARCH

version 1.1 (20.1 KB) by Jonas Lundgren

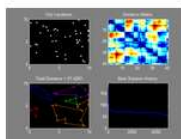
Heuristic method for the Traveling Salesman Problem (TSP)

Overview

Functions

A number of nearest neighbour tours are generated from randomly selected starting points. Each tour is improved by 2-opt heuristics (pairwise exchange of edges) and the best result is selected.

<https://www.mathworks.com/matlabcentral/fileexchange/35178-tspsearch>



Multiple Traveling Salesmen Problem - Genetic Algorithm

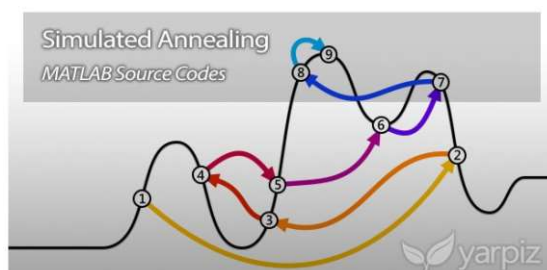
version 1.3 (13.7 KB) by Joseph Kirk

Finds a near-optimal solution to a M-TSP using a GA

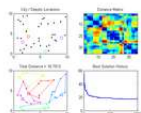
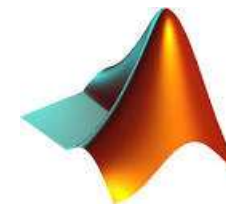
MTSP_GA Multiple Traveling Salesmen Problem (M-TSP) Genetic Algorithm (GA)

Finds a (near) optimal solution to the M-TSP by setting up a GA to search for the shortest route (least distance needed for the salesmen to travel to each city exactly once and return to their starting locations)

<https://www.mathworks.com/matlabcentral/fileexchange/19049-multiple-traveling-salesmen-problem-genetic-algorithm>



The open-source MATLAB implementation of Simulated Algorithm, which is used to solve the Traveling Salesman Problem (TSP). In addition to standard version of SA, implementation of a Population-based Simulated Annealing is also provided within the download package.



MDMTSPV_GA - Multiple Depot Multiple Traveling Salesmen Problem solved by Genetic Algorithm

version 1.0 (5.44 KB) by [Elad Kivelevitch](#)

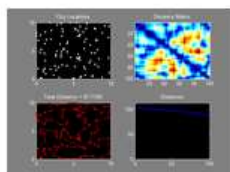
Genetic Algorithm Solution to the Multiple Depots, MTSP, with Variable number of salesmen

Finds a (near) optimal solution to a variation of the M-TSP (that has a variable number of salesmen) by setting up a GA to search for the shortest route (least distance needed or the salesmen to travel to each city exactly once and return to their starting locations). The salesmen originate from a set of fixed locations, called depots.

This algorithm is based on Joseph Kirk's MTSPV_GA, but adds the following functionality:

1. Depots at which each salesman originates and ends its tour.
2. Two possible cost functions, that allow to find minimum sum of all tour lengths (as in the original version) and to find the minimum longest tour. The latter problem is sometimes called MinMaxMDMTSP.

<https://www.mathworks.com/matlabcentral/fileexchange/31814-mdmtspv-ga-multiple-depot-multiple-traveling-salesmen-problem-solved-by-genetic-algorithm>



Traveling Salesman Problem - Nearest Neighbor

version 1.3 (8.44 KB) by [Joseph Kirk](#)

Finds a near-optimal solution to a TSP using Nearest Neighbor (NN)

TSP_NN Traveling Salesman Problem (TSP) Nearest Neighbor (NN) Algorithm

The Nearest Neighbor algorithm produces different results depending on which city is selected as the starting point. This function determines the Nearest Neighbor routes for multiple starting points and returns the best of those routes

<https://www.mathworks.com/matlabcentral/fileexchange/21297-traveling-salesman-problem-nearest-neighbor>



In this post we are going to share with you, the complete and open source implementation of ACO in MATLAB. The ACO is implemented to solve the following problems, in three different projects:

- ✓ Traveling Salesman Problem (TSP)
- ✓ Quadratic Assignment Problem (QAP)
- ✓ Binary Knapsack Problem

<http://yarpiz.com/53/ypea103-ant-colony-optimization>



LKH_TSP

A set of tools to solve TSP problems using the LKH solver

What is LKH

LKH is an effective implementation of the Lin-Kernighan heuristic for solving the Traveling Salesman Problem. The Lin-Kernighan solver (also called the Lin-Kernighan-Helsgaun solver) is among the most efficient solvers of the TSP and it is employing the concept of k-opt moves. An implementation of this solver is found online following this link:

<http://www.akira.ruc.dk/~keld/research/LKH/>

From this site you can download and compile the code following the relevant instructions. Considering that this process is done correctly, this repo provides simple tools to easily invoke this functionality.

Python interface Within *python* a python script called *InvokeLKH.py* interfaces a compiled version of the LKH TSP Solver and exports the solution in the form of a file. To run the script:

```
$ python InvokeLKH.py
```

MATLAB interface Within *matlab* a matlab function called *LKH_TSP.m* interfaces a compiled version of the LKH TSP Solver and exports the solution to its output argument. The function syntax is:

```
TSPsolution = LKH_TSP(CostMatrix,pars_struct,fname_tsp,LKHdir,TSPLIBdir)
```

Feel free to use these simple functions which are released mostly to motivate those working on TSP to use the LKH solver.

A set of tools to solve TSP problems using the LKH solver

6 commits		1 branch	0 releases	1 contributor
Branch: master		New pull request		Find file Clone or download
kostas-alexis Update LKH_TSP.m		Update LKH_TSP.m		Latest commit 8c14fe2 on Nov 8 2015
matlab		Update InvokeLKH.py		2 years ago
python		Update README.md		2 years ago
README.md				2 years ago



tsp 0.0.1

'tsp' is a package for Traveling Salesman Problem.

tsp is a package for Traveling Salesman Problem. Using Concorde. <http://www.math.uwaterloo.ca/tsp/concorde/>

```
t = tsp([(0,0), (0,1), (1,0), (1,1)])
t.solve()
print(t.result)
>>>
[0, 1, 3, 2]
```

<https://pypi.python.org/pypi/tsp/0.0.1>

Solves the TSP using multithreaded ACO method

trevlovet Update README		Latest commit 354d1e6 on Oct 30 2012
README	Update README	5 years ago
ant.py	Everything you need	6 years ago
antcolony.py	Everything you need	6 years ago
antgraph.py	Everything you need	6 years ago
anttsp.py	Everything you need	6 years ago
citiesAndDistances.pickled	Everything you need	6 years ago

<https://github.com/trevlovet/Python-Ant-Colony-TSP-Solver>

TSP solver

Advanced algorithms practice project of MLDM first year students for TSP problem ##Description The objective of this project is to implement some algorithms for solving Traveling Salesman Problem (TSP) and to provide an experimental study of their running time and the quality of the outputs as well. ##Algorithms Included The algorithms developed for solving the TSP problem are:

1. Brute-force approach
2. Branch-and-Bound approach
3. 2-opt (edge swapping)
4. Minimum Spanning Tree approximation
5. Greedy approach and Iterative greedy
6. Randomized approach
7. Genetic algorithm
8. Evolutionary algorithm (Hillclimbing based)

https://github.com/thovo/TSP_solver

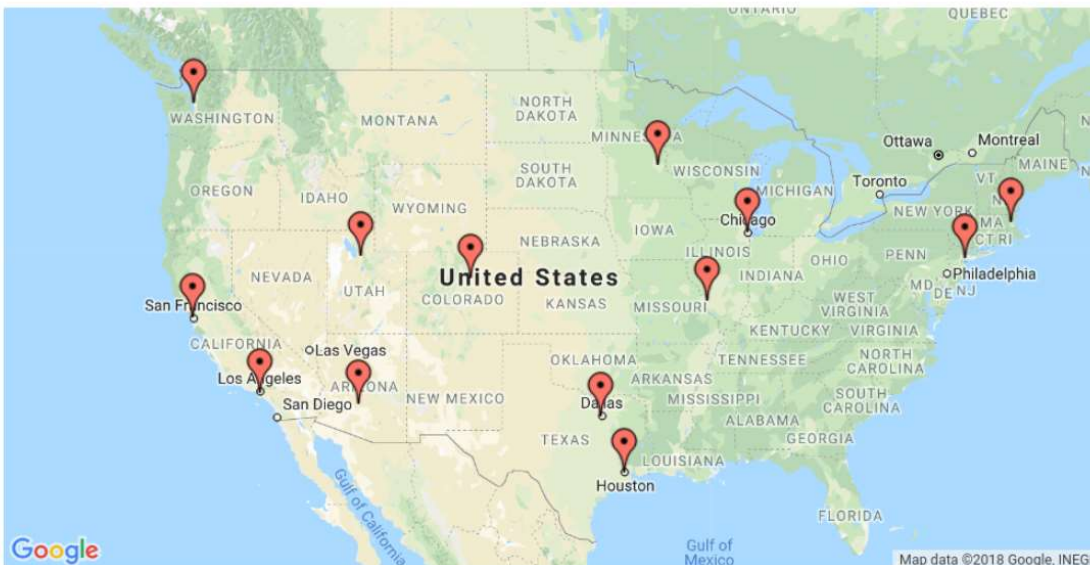
[Home](#)
[Installation](#)
[Guides](#)
[Reference](#)
[Examples](#)
[Support](#)

Filter

Assignment
Packing
Scheduling
Routing
Overview
Traveling Salesperson Problem
Vehicle Routing Problem
Capacity Constraints
Pickups and Deliveries
Time Window Constraints
Dimensions
Resource Constraints
Penalties and Dropping Visits
Common Routing Tasks
Routing Options
Google Directions API

Traveling Salesperson Problem

This section presents an example that shows how to solve the Traveling Salesperson Problem (TSP) for the locations shown on the map below.



Google OR Tools — A Guide

<https://medium.com/google-or-tools/google-or-tools-a-guide-39f439a5cd0f>



PyConcorde



PyConcorde is a Python wrapper around the [Concorde TSP solver](#).

PyConcorde allows you to compute solutions to the Traveling Salesman Problem with just a few lines of Python code. It uses the state-of-the-art Concorde solver and provides a convenient Python layer around it.





Concorde Home

Concorde TSP Solver

Concorde is a computer code for the symmetric traveling salesman problem (TSP) and some related network optimization problems. The code is written in the ANSI C programming language and it is available for academic research use; for other uses, contact William Cook for licensing options.

Concorde's TSP solver has been used to obtain the optimal solutions to all 110 of the TSPLIB instances, the largest having 85,900 cities.

The Concorde callable library includes over 700 functions permitting users to create specialized codes for TSP-like problems. All Concorde functions are thread-safe for programming in shared-memory parallel environments; the main TSP solver includes code for running over networks of UNIX workstations.

Concorde now supports the QSOPT linear programming solver. Executable versions of Concorde with qsopt for Linux and Solaris are now available.

Hans Mittelmann has created a NEOS Server for Concorde, allowing users to solve TSP instances online.

Home | Concorde Home

Back

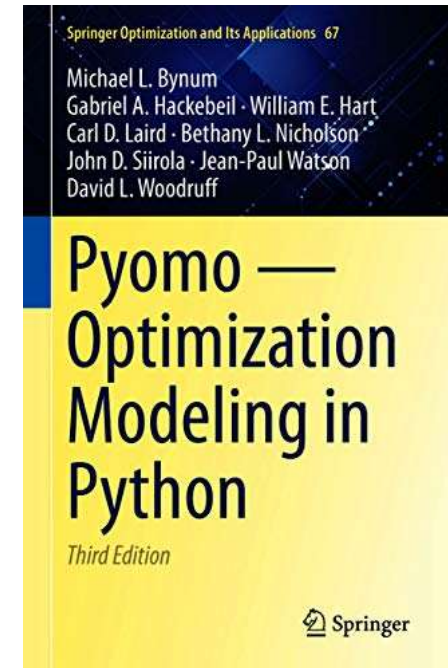
Last Updated: October 2020

Pyomo Overview



Pyomo is a Python-based open-source software package that supports a diverse set of optimization capabilities for formulating and analyzing optimization models. Pyomo can be used to define symbolic problems, create concrete problem instances, and solve these instances with standard solvers. Pyomo supports a wide range of problem types, including:

- Linear programming
- Quadratic programming
- Nonlinear programming
- Mixed-integer linear programming
- Mixed-integer quadratic programming
- Mixed-integer nonlinear programming
- Mixed-integer stochastic programming
- Generalized disjunctive programming
- Differential algebraic equations
- Mathematical programming with equilibrium constraints
- Constraint programming



●● Model and solution of the Traveling Salesman Problem with Python and Pyomo

<https://medium.com/analytics-vidhya/model-and-solution-of-the-traveling-salesman-problem-with-python-and-pyomo-db45f2631e8c>

LP Travelling Salesman Problem in Gurobi (with Python)

Implementation of several LP formulations to solve TSP in LP, the following formulations are implemented

- Dantzig, Fulkerson and Johnson (DFJ)
- Miller-Tucker-Zemlin (MTZ)
- Flow
- Dantzig Step

*DFJ is implemented to only include subtour constraints if they are not redundant.

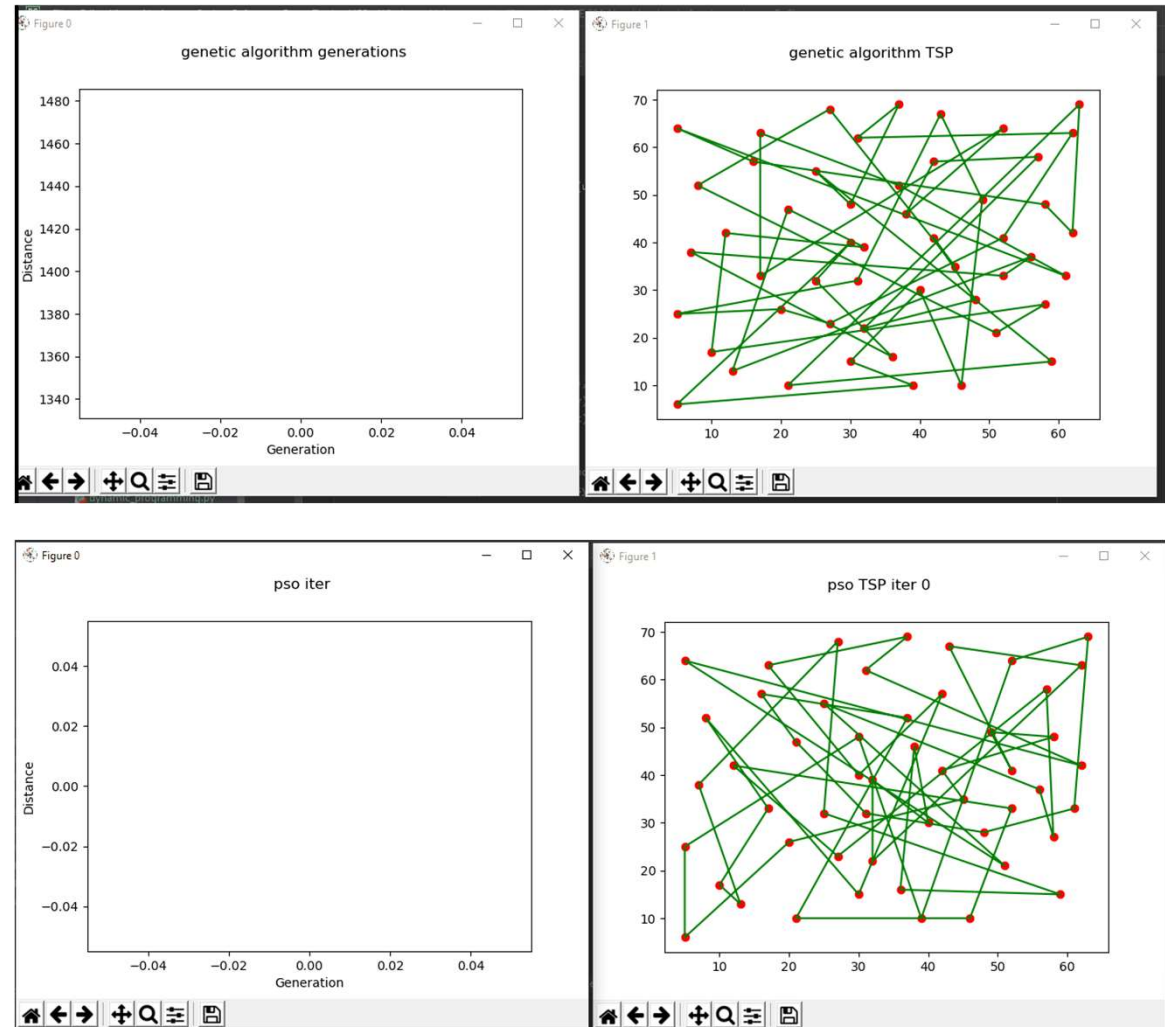
<https://github.com/Arthod/LP-tsp-gurobi>

Solving the Travelling Salesman Problem in Python

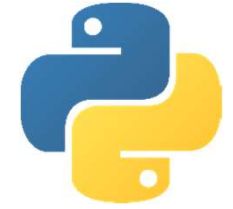


Implemented techniques

- Genetic Algorithm
- Simulated Annealing
- PSO: Particle Swarm Optimization
- Divide and conquer
- Dynamic Programming
- Greedy
- Brute Force

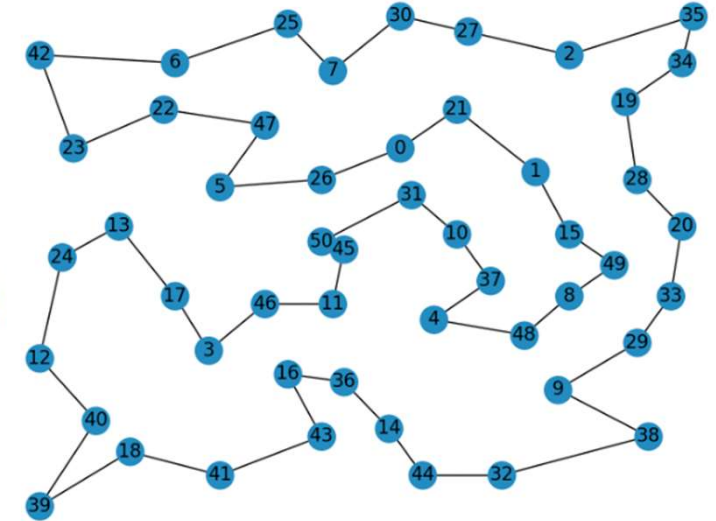


tsp-solver



metaheuristics for traveling salesman problem (TSP)

- `tsp_ls_naive.py` Local Search (LS)
- `tsp_ls_nblast.py` Local Search with Neighbor-List (LS-NL)
- `tsp_mls.py` Random Multi-start Local Search (MLS)
- `tsp_ils.py` Iterated Local Search (ILS)
- `tsp_ils_fls.py` Iterated Local Search with Fast Local Search (ILS-FLS)
- `tsp_ils_imp.py` Improved ILS-FLS (ILS+)
- `tsp_gls.py` Guided Local Search (GLS)
- `tsp_gls_fls.py` Guided Local Search with FLS (GLS-FLS)
- `tsp_grasp.py` Greedy Randomized Adaptive Search Procedure (GRASP)
- `tsp_sa.py` Simulated Annealing (SA)
- `tsp_tabu_rule1.py` Tabu Search with Rule1 (TS1)
- `tsp_tabu_rule2.py` Tabu Search with Rule2 (TS2)
- `tsp_ma.py` Memetic Algorithm (MA)
- `tsp_ma_fls.py` Memetic Algorithm with FLS (MA-FLS)





pypi v0.6.6 build passing codecov 92% license MIT python >=3.5 platform windows | linux | macos Forks 851
downloads 113k discussions

Swarm Intelligence in Python

(Genetic Algorithm, Particle Swarm Optimization, Simulated Annealing, Ant Colony Algorithm, Immune Algorithm, Artificial Fish Swarm Algorithm in Python)

- **Documentation:** <https://scikit-opt.github.io/scikit-opt/#/en/>
- **文档:** <https://scikit-opt.github.io/scikit-opt/#/zh/>
- **Source code:** <https://github.com/guofei9987/scikit-opt>
- **Help us improve scikit-opt** <https://www.wjx.cn/jq/50964691.aspx>

Type to search

Document

install

Features

Feature1: UDF

feature2: continue to run

feature3: 4-ways to accelerate

feature4: GPU computation

Quick start

1. Differential Evolution

2. Genetic Algorithm

3. PSO(Particle swarm optimization)

4. SA(Simulated Annealing)

5. ACA (Ant Colony Algorithm) for tsp

6. immune algorithm (IA)

7. Artificial Fish Swarm Algorithm (AFSA)

Projects using scikit-opt

About parameters

More Genetic Algorithm

More Particle Swarm Optimization

More Simulated Annealing

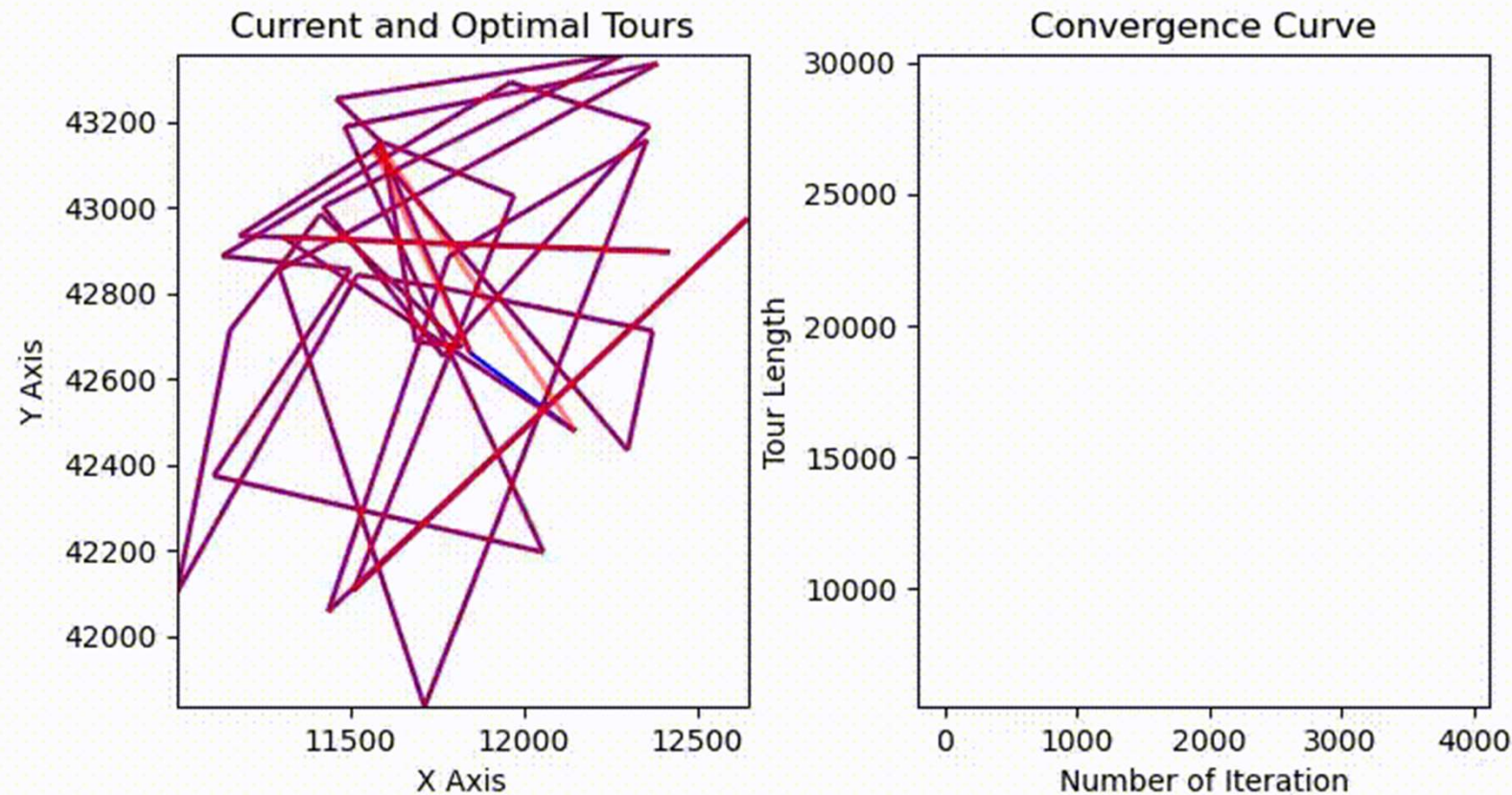
Curve fitting

Speed Up



tsp-meta-heuristic

Python implementation of Tabu Search (TB), Genetic Algorithm (GA), and Simulated Annealing (SA) solving Travelling Salesman Problem (TSP). Term project of Intelligent Optimization Methods, UCAS course [070105M05002H](#).





TSP – Infrastructure for the Traveling Salesperson Problem

Michael Hahsler
Southern Methodist University

Kurt Hornik
Wirtschaftsuniversität Wien

Abstract

The traveling salesperson problem (also known as traveling salesman problem or TSP) is a well known and important combinatorial optimization problem. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. Despite this simple problem statement, solving the TSP is difficult since it belongs to the class of NP-complete problems. The importance of the TSP arises besides from its theoretical appeal from the variety of its applications. Typical applications in operations research include vehicle routing, computer wiring, cutting wallpaper and job sequencing. The main application in statistics is combinatorial data analysis, e.g., reordering rows and columns of data matrices or identifying clusters. In this paper we introduce the R package **TSP** which provides a basic infrastructure for handling and solving the traveling salesperson problem. The package features S3 classes for specifying a TSP and its (possibly optimal) solution as well as several heuristics to find good solutions. In addition, it provides an interface to *Concorde*, one of the best exact TSP solvers currently available.

<https://cran.r-project.org/web/packages/TSP/vignettes/TSP.pdf>

Package ‘TSP’

February 22, 2017

Type Package

Title Traveling Salesperson Problem (TSP)

Version 1.1-5

Date 2017-02-21

Description Basic infrastructure and some algorithms for the traveling salesperson problem (also traveling salesman problem; TSP). The package provides some simple algorithms and an interface to the Concorde TSP solver and its implementation of the Chained-Lin-Kernighan heuristic. The code for Concorde itself is not included in the package and has to be obtained separately.

<https://cran.r-project.org/web/packages/TSP/TSP.pdf>



TSP Algorithm Selection

The **Travelling Salesperson Problem** (TSP) is arguably the most prominent NP-hard combinatorial optimisation problem. Given a set of n cities and pairwise distances between those, the objective in the TSP is to find the shortest round-trip or tour through all cities, i.e., a sequence in which every city is visited exactly once, the start and end cities are identical, and the total length of the tour is minimal. The Euclidean TSP has important applications, e.g., in the fabrication of printed circuit boards as well as in transportation and logistics. We aim at constructing an instance-based algorithm selection model in order to improve the current state-of-the-art solver.

- **R package netgen** [[GitHub](#), [CRAN](#)]

Methods for generating random or clustered networks in order to benchmark algorithms for combinatorial optimization problems on graphs, e.g. the Travelling-Salesperson-Problem (TSP) or the Vehicle-Routing-Problem (VRP). Furthermore, this package contains methods for morphing networks, importing from and exporting into the TSPLib format, as well as various visualization techniques.

- **R package salesperson** [[GitHub](#)]

Comprehensive collection of functions for solving and analyzing the symmetric Traveling Salesperson Problem (TSP) by means of instance characteristics, frequently termed instance features.

- **R package tspgen** [[GitHub](#)]

Sophisticated benchmark problem generator, which produces TSP instances by a sequential random process adopting a variety of problem tailored mutations.

Additional material



[European Conference on Evolutionary Computation in Combinatorial Optimization \(Part of EvoStar\)](#)

..... EvoCOP 2019: [Evolutionary Computation in Combinatorial Optimization](#) pp 99-114 | [Cite as](#)

Rigorous Performance Analysis of State-of-the-Art TSP Heuristic Solvers

Authors

Authors and affiliations

Paul McMenemy , Nadarajen Veerapen, Jason Adair, Gabriela Ochoa

Conference paper

First Online: 28 March 2019

345

Downloads

Part of the [Lecture Notes in Computer Science](#) book series (LNCS, volume 11452)

Abstract

Understanding why some problems are better solved by one algorithm rather than another is still an open problem, and the symmetric Travelling Salesperson Problem (TSP) is no exception. We apply three state-of-the-art heuristic solvers to a large set of TSP instances of varying structure and size, identifying which heuristics solve specific instances to optimality faster than others. The first two solvers considered are variants of the multi-trial Helsgaun's Lin-Kernighan Heuristic (a form of iterated local search), with each utilising a different form of Partition Crossover; the third solver is a genetic algorithm (GA) using Edge Assembly Crossover. Our results show that the GA with Edge Assembly Crossover is the best solver, shown to significantly outperform the other algorithms in 73% of the instances analysed. A comprehensive set of features for all instances is also extracted, and decision trees are used to identify main features which could best inform algorithm selection. The most prominent features identified a high proportion of instances where the GA with Edge Assembly Crossover performed significantly better when solving to optimality.

Performance Analysis of Local Optimization Algorithms in Traveling Salesman Problem

 894



Abstract:

There are plenty of intelligence algorithms and heuristic algorithms for TSP (Traveling Salesman Problem). In this paper, local optimization algorithm which is a good representative of heuristic algorithms was analyzed. The performance of 2-opt (optimization), 3-opt and 4-opt were analyzed and compared through experiments. To reduce running time and improve their feasibility, a modification was made on 3-opt and 4-opt. Ant colony optimization as a good representative of intelligence algorithm was combined with k -opt to analyze. The results provide reference to application of k -opt and designing optimization algorithms for TSP in future.

Info:

Periodical:	Advanced Materials Research (Volumes 846-847)
Edited by:	Q. Lu and C.G. Zhang
Pages:	1364-1367
DOI:	https://doi.org/10.4028/www.scientific.net/AMR.846-847.1364
Citation:	Cite this paper

Is > ACM Journal of Experimental Algorithmics > Vol. 23 > Analysis of a High-Performance TSP Solver on the GPU

RESEARCH-ARTICLE

Analysis of a High-Performance TSP Solver on the GPU



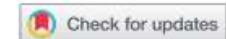
Authors: [Jeffrey A. Robinson](#), [Susan V. Vrbisky](#), [Xiaoyan Hong](#), [Brian P. Eddy](#) [Authors Info & Affiliations](#)

Publication: ACM Journal of Experimental Algorithmics • January 2018 • Article No.: 1.1

• <https://doi.org/10.1145/3154835>

0 676





Performance Analyses of Nature-inspired Algorithms on the Traveling Salesman's Problems for Strategic Management

Julius Beneoluchi Odili^a, Mohd Nizam Mohmad Kahar^a, A. Noraziah^{a,b}, M. Zarina^c and Riaz Ul Haq^a

^aFaculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Kuantan, Malaysia; ^bIBM Centre of Excellence, Universiti Malaysia Pahang, Kuantan, Malaysia; ^cFaculty of Informatic and Computing, Universiti Sultan Zainal Abidin, Gong Badak, Malaysia

ABSTRACT

This paper carries out a performance analysis of major Nature-inspired Algorithms in solving the benchmark symmetric and asymmetric Traveling Salesman's Problems (TSP). Knowledge of the workings of the TSP is very useful in strategic management as it provides useful guidance to planners. After critical assessments of the performances of eleven algorithms consisting of two heuristics (Randomized Insertion Algorithm and the Honey Bee Mating Optimization for the Travelling Salesman's Problem), two trajectory algorithms (Simulated Annealing and Evolutionary Simulated Annealing) and seven population-based optimization algorithms (Genetic Algorithm, Artificial Bee Colony, African Buffalo Optimization, Bat Algorithm, Particle Swarm Optimization, Ant Colony Optimization and Firefly Algorithm) in solving the 60 popular and complex benchmark symmetric Travelling Salesman's optimization problems out of the total 118 as well as all the 18 asymmetric Travelling Salesman's Problems test cases available in TSPLIB91. The study reveals that the African Buffalo Optimization and the Ant Colony Optimization are the best in solving the symmetric TSP, which is similar to intelligence gathering channel in the strategic management of big organizations, while the Randomized Insertion Algorithm holds the best promise in asymmetric TSP instances akin to strategic information exchange channels in strategic management.

KEYWORDS

Decision-making; logistics management; nature-inspired algorithms; performance analysis; strategic planning; traveling salesman's problems

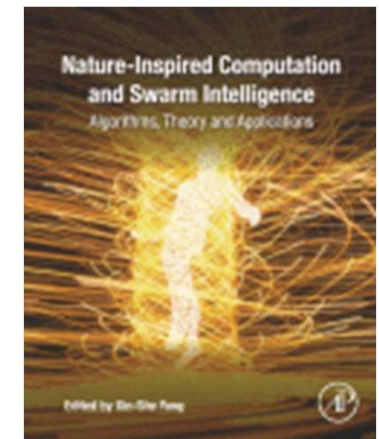
Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics

Eneko Osaba^a, Xin-She Yang^b, Javier Del Ser^{a,c}

^aTECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain

^bMiddlesex University London, School of Science and Technology, London, United Kingdom

^cUniversity of the Basque Country (UPV/EHU), Bilbao, Spain



CONTENTS

9.1	Introduction.....	136
9.2	Problem statement	137
9.3	Recent advances in the traveling salesman problem.....	139
9.3.1	TSP and genetic algorithms.....	139
9.3.2	TSP and simulated annealing.....	140
9.3.3	TSP and tabu search	140
9.3.4	TSP and ant colony optimization	141
9.3.5	TSP and particle swarm optimization	142
9.3.6	TSP and bat algorithm	142
9.3.7	TSP and firefly algorithm	143
9.3.8	TSP and cuckoo search.....	143
9.3.9	TSP and artificial bee colony	143
9.3.10	TSP and imperialist competitive algorithm	144
9.3.11	TSP and other nature-inspired metaheuristics	145
9.4	Novelty search	145
9.5	Proposed bio-inspired methods	147
9.5.1	Bat algorithm.....	149
9.5.2	Firefly algorithm	149
9.5.3	Particle swarm optimization	150
9.6	Experimentation and results	150
9.7	Research opportunities and open challenges	152
9.8	Conclusions	154
	Acknowledgment.....	154
	References.....	154



Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor



Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs

Güneş Erdoğan^{a,*}, Maria Battarra^b, Gilbert Laporte^c, Daniele Vigo^d

^a Industrial Engineering Department, Özyeğin University, Kuşbakişı Cad. 2, 34662 Altunizade, İstanbul, Turkey

^b School of Mathematics, University of Southampton, Southampton SO17 1BJ, UK

^c Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, Canada H3T 2A7

^d DEIS, University of Bologna, via Venezia 52, 47521 Cesena, Italy

ARTICLE INFO

Available online 22 July 2011

Keywords:

Traveling salesman problem

Handling cost

Pickup and delivery

Metaheuristics

ABSTRACT

This paper studies the Traveling Salesman Problem with Pickups, Deliveries, and Handling Costs. The subproblem of minimizing the handling cost for a fixed route is analyzed in detail. It is solved by means of an exact dynamic programming algorithm with quadratic complexity and by an approximate linear time algorithm. Three metaheuristics integrating these solution methods are developed. These are based on tabu search, iterated local search and iterated tabu search. The three heuristics are tested and compared on instances adapted from the related literature. The results show that the combination of tabu search and exact dynamic programming performs the best, but using the approximate linear time algorithm considerably decreases the CPU time at the cost of slightly worse solutions.

© 2011 Published by Elsevier Ltd.



Hybrid metaheuristics in combinatorial optimization: A survey

Christian Blum^{a,*}, Jakob Puchinger^b, Günther R. Raidl^c, Andrea Roli^d

^a ALBCOM Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain

^b Mobility Department, Austrian Institute of Technology, Vienna, Austria

^c Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

^d Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Alma Mater Studiorum, Università di Bologna, Campus of Cesena, Italy

ARTICLE INFO

Article history:

Received 29 August 2010

Received in revised form 12 February 2011

Accepted 24 February 2011

Available online 15 March 2011

Keywords:

Hybrid metaheuristics

Combinatorial optimization

Mathematical programming

Constraint programming

Local search

ABSTRACT

Research in metaheuristics for combinatorial optimization problems has lately experienced a noteworthy shift towards the hybridization of metaheuristics with other techniques for optimization. At the same time, the focus of research has changed from being rather algorithm-oriented to being more problem-oriented. Nowadays the focus is on solving the problem at hand in the best way possible, rather than promoting a certain metaheuristic. This has led to an enormously fruitful cross-fertilization of different areas of optimization. This cross-fertilization is documented by a multitude of powerful hybrid algorithms that were obtained by combining components from several different optimization techniques. Hereby, hybridization is not restricted to the combination of different metaheuristics but includes, for example, the combination of exact algorithms and metaheuristics. In this work we provide a survey of some of the most important lines of hybridization. The literature review is accompanied by the presentation of illustrative examples.

© 2011 Elsevier B.V. All rights reserved.

Quote

There are no facts, **only** interpretations.

Friedrich Wilhelm Nietzsche (1844-1900)

German philosopher, cultural critic, composer, poet, and philologist whose work has exerted a profound influence on modern intellectual history.

