

Chapter 2

On the Effect of Adding Nodes to TSP Instances: An Empirical Analysis

Gloria Cerasela Crişan, Elena Nechita and Vasile Palade

Abstract Our human society is experiencing complex problems nowadays, which require large amounts of computing resources, fast algorithms and efficient implementations. These real-world problems generate new instances for the classical, academic problems as well as new data collections that can be used for assessing the available solving packages. This paper focuses on the Traveling Salesman Problem, which is one of the most studied combinatorial optimization problems, with many variants and broad applications. In order to allow a smooth integration with the current Geographic Information Systems (GIS) technologies, the instances described in this work are specified by geographic coordinates, and they use the orthodromic distance. A sequence of similar instances is defined, and the characteristics of the state-of-the-art exact solver results on these instances are presented and discussed.

Keywords Combinatorial optimization • Traveling Salesman Problem • Exact algorithms • Metaheuristics • Orthodromic distance

2.1 Introduction

The Traveling Salesman Problem (TSP) has been intensively studied by researchers all over the world. A search process over the Web of Science citation indexing service, maintained by Thomson Reuters, performed for the period 2000–2015 with the keywords “Traveling Salesman Problem”, returns more than 7000 entries.

G.C. Crişan (✉) • E. Nechita

Vasile Alecsandri University of Bacău, Bacău, Romania
e-mail: ceraselacrisan@ub.ro

E. Nechita
e-mail: enechita@ub.ro

V. Palade
Coventry University, Coventry, UK
e-mail: vasile.palade@coventry.ac.uk

However, the work developed by the scientific community, related to the TSP and its variants, is definitely much more extensive. Discussions on the TSP are present in many processes, starting with the educational ones, offered by the higher education institutions that include Computer Science degrees in their curricula and ending with the research developed within the most famous world's institutes in Artificial Intelligence.

The attention given to this combinatorial optimization problem comes from its significance for both theoretical advances and real-world applications. In addition, the TSP is one of the standard test problems used in the performance analysis of discrete optimization algorithms.

The goal of this paper is to empirically study the behavior of an exact TSP solver on a set of very close real-world instances. This work considers that two TSP instances are similar if one is obtained by adding a new node to the other, together with the costs of the corresponding generated edges. We used real data, downloaded from free online repositories. In order to comply with the large geographic area where data are sampled from, we used the orthodromic (great-circle) distances as travel costs. This research is a combination of approaches, since it uses the modern GIS format in order to define a structured collection of TSP instances. This set is solved with a state-of-the-art academic exact solver, and the results are interpreted using the re-optimization framework. Moreover, the solver uses a combination of techniques.

The next section presents a brief overview of several TSP variants and approaches. Section 2.3 describes the design and the implementation of an experiment for studying the influence of adding one single node into a medium-sized TSP instance. Section 2.4 contains the results of our computational study and their interpretations. Section 2.5 summarizes the conclusions of this research.

2.2 TSP—The Problem and Its Variants

The *general TSP* considers a set of n vertices and the nonnegative pairwise costs associated to the edges between those vertices. The objective of the problem is to find the minimum cost tour passing exactly once through each vertex. Formally, the TSP is defined as follows.

Let $G = (V, A)$ be a complete graph, with $V = \{v_1, \dots, v_n\}$ the set of n vertices and $A = \{(v_i, v_j) \mid v_i, v_j \in V\}$ the set of arcs. Let $C = (c_{ij})$ be the cost matrix associated with A , with c_{ij} being the cost of (v_i, v_j) . The TSP consists in determining the least cost Hamiltonian tour of G [1].

It is worth noting that the general TSP sets up no restrictions over the cost function. Since the Hamiltonian circuit decision problem is NP-complete [2], the TSP is also NP-complete.

The TSP is called *symmetric* if $c_{ij} = c_{ji}$ for all $v_i, v_j \in V$, otherwise it is called *asymmetric*. Therefore, the symmetric TSP is defined on a complete undirected

graph, while the asymmetric TSP (ATSP) is defined on a complete directed one. C satisfies the triangle inequality if and only if $c_{ij} + c_{jk} \geq c_{ik}$ for all $v_i, v_j, v_k \in V$. Hereinafter, if not specified otherwise, the TSP denotes the symmetric TSP.

The *metric TSP* is the TSP whose vertices lie in a metric space (all edge costs are symmetric and fulfil the triangle inequality). The *Euclidean TSP* is a subcase of the metric TSP. In this case, the vertices are placed in R^d , $d \geq 2$ and the cost is defined using the l_d norm [3]. The most addressed variant of the Euclidean TSP is that for $d = 2$, due to its interpretations and applicability in real-life situations.

The *2D Euclidean TSP* considers a set of vertices which correspond to n locations in R^2 . The objective in this common variant is to find the shortest tour through all locations. The Vehicle Routing Problem (VRP) also derives from the 2D Euclidean TSP [4].

Complexity studies on the TSP [5, 6] show that the problem is computationally difficult. The exact optimization of the general TSP is NP-hard [7] and so is the Euclidean TSP [8]. Nevertheless, in time, researchers have studied significant variants and instances, which arise in conjunction with the applications that can be modelled with the TSP as mathematical support. For many of the instances, the hardness results may not necessarily apply [6]. The following subsections present several TSP variants and considerations related to their complexity.

2.2.1 The Traveling Salesman Problem—Variants and Their Complexity

Applegate et al. [9] provide an extensive view over the advances in the TSP, until 2011. Our survey is conducted by the need of approaching the particularly generated TSP instances, in connection with the possibility to reuse the knowledge about the optimum results already determined for previously studied instances. Nowadays, new TSP instances may arise from various modern applications. Industry, Logistics, Transportations face specific scenarios and need specific models and simulations. Uncertainty, randomness and technological advances (such as the use of drones, reminded below) made us look towards very new versions of the TSP.

The Probabilistic Traveling Salesman Problem (PTSP) is a more difficult variant of the classical TSP. It was introduced by Jaillet [10] and came from the necessity to adopt a model that takes into account the random real life phenomena [11]: for many delivery companies that perform pickup and delivery, not all the customers require daily visits. In the PTSP, a visiting probability between 0 and 1 is assigned to each vertex. On a daily basis, according to its visiting probabilities, each vertex requests a visit or announces the visit skipping. The PTSP solution is a complete a priori tour of minimal expected length, which gives the order for the nodes that requested a visit on a certain day. The other nodes are skipped [12]. The PTSP formulation, given in [10], is the following:

Let T be an a priori PTSP tour through n vertices of a given graph G , where each vertex i has a visiting probability p_i independently of the others. Let d_{ij} be the distance between i and j . Without any loss of generality, we may assume that the a priori tour T is $(1, 2, \dots, n, 1)$. The problem is to find that complete a priori tour which minimizes the expected length $E[L_T]$:

$$E[L_T] = \sum_{i=1}^n \sum_{j=i+1}^n d_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1 - p_k) + \sum_{i=1}^n \sum_{j=1}^{i-1} d_{ji} p_i p_j \prod_{k=i+1}^n (1 - p_k) \prod_{l=1}^{j-1} (1 - p_l)$$

Obviously, the TSP is a special case of the PTSP, in which all the n vertices have $p_i = 1$. For instances with up to 50 vertices, branch-and-bound algorithms [13] and exact branch-and-cut algorithms [14] have been proposed to find the PTSP optimal solutions. However, for larger instances, most approaches to the PTSP are also based on heuristics and metaheuristics.

The Traveling Salesman Problem with Drone (TSP-D) is a new variant of the TSP [15]. Over the last years, many engineers have studied the possibility of using drones in combination with vehicles, to support deliveries and make them more cost-effective [16]. The idea involves both assignment decisions and routing decisions and it has generated the TSP-D.

Formally, the TSP-D is modeled in a graph $G = (V, A)$, where the node v_0 represents a depot and the n nodes v_1, \dots, v_n are the customer locations. The edges $e_{ij} = (v_i, v_j)$ connect the nodes v_i, v_j and c_{ij} is the driving time that a vehicle needs to commute from v_i to v_j or vice versa (therefore, the triangle inequality holds). The objective of the TSP-D is to find the shortest tour to serve all customers either by vehicle or by drone, in terms of time [15].

The TSP-D is related to the Covering Salesman Problem (CSP) introduced in 1989 by Current and Schilling [17], where the aim is to find the shortest tour of a subset of given nodes, while every node not belonging to the tour is within a predefined covering distance of a node on the tour. The TSP-D differs from CSP because the nodes outside the tour have to be visited by drones, and these have to be synchronized with the vehicles. This feature lines up the TSP-D with the class of the Vehicle Routing Problems (VRPs) [18] and also with the Truck and Trailer Routing Problem (TTRP) [19]. Moreover, the TSP-D is similar to the so-called “Flying Sidekick Traveling Salesman Problem”, proposed in 2015 by Murray and Chu [20].

The TSP-D is also NP-hard. Under the assumptions that (a) the drones return to the truck after each delivery and (b) the pickups from the truck always take place at a customer location or at the depot, a solution would be a pair of tours (R, D) , where R is a vehicle route from v_0 to v_0 , together with a drone route D which includes all the customers that are visited by both truck and drone [15].

Introduced in 2004, the **one-commodity pickup-and-delivery Traveling Salesman Problem** (1-PDTSP) is a TSP plus the following additional constraints [21]: one specific city is the depot for a vehicle (with a fixed upper limit capacity) which visits the customers divided into two groups, according to the required

service: delivery or pickup. Each delivery customer demands a given amount of a product, while each pickup customer provides a given amount of that product. Any amount collected from the pickup clients can be transferred to the delivery customers. The solution is a Hamiltonian tour through the cities, without ever exceeding the vehicle capacity. Several studies show that finding feasible 1-PDTSP solutions is much more complicated than finding good heuristic solutions for TSP [21, 22]. Combined approaches, involving the TSP heuristics and the branch-and-cut algorithms provide good solutions for the large instances, and also for the classical TSP with pickup and delivery (TSPPD) [23].

2.2.2 The Traveling Salesman Problem—Approaches

Apart from the applications directly related to (vehicle) routing, literature describes situations that can be modeled by the TSP, coming from various domains: computer wiring and dashboard design, hole punching and wallpaper cutting, applications in crystallography and in polyhedral theory [24]. Most of them are large scale applications and cannot be tackled using exact algorithms. Instead, heuristics are used to provide solutions, which can be considered very good in terms of computing time and/or computer resources. A short review over the most important exact and approximate algorithms is given in the following.

The Exact Methods for the TSP are connected to the developments in the field of the Integer Linear Programming (ILP). In this framework, the TSP formulation of Dantzig, Fulkerson and Johnson [25], further on referred as with DFJ, comes from 1954.

Let x_{ij} be a binary variable associated to the arc (v_i, v_j) , with $x_{ij} = 1$ if and only if (v_i, v_j) belongs to the optimal solution. The DFJ formulation is as follows:

Minimize

$$\sum_{\substack{i,j=1 \\ i \neq j}}^n c_{ij}x_{ij} \quad (2.1)$$

subject to

$$\sum_{j=1}^n x_{ij} = 1, \quad i = 1, \dots, n \quad (2.2)$$

$$\sum_{i=1}^n x_{ij} = 1, \quad j = 1, \dots, n \quad (2.3)$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad S \subset V, \quad 2 \leq |S| \leq n-2 \quad (2.4)$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \dots, n, \quad i \neq j \quad (2.5)$$

The objective function (2.1) describes the cost of the optimal tour. The constraints (2.2) and (2.3) specify that every vertex is entered exactly once and left exactly once (degree constraints). The forbiddance of tours on the subsets S of less than n vertices is given in (2.4) (subtour elimination constraints); as well, S cannot be disconnected (there must be at least one arc pointing from S to its complement). The binary conditions on the variables are described by (2.5). Under $n(n-1)$ variables, $2n$ degree constraints and $2^n - 2n - 2$ subtour elimination constraints, the DFJ cannot be solved by ILP code even for moderate values of n . In order to approach the larger problems in the context of mathematical programming, alternative formulations have been proposed by relaxing the constraints (2.4), thus resulting assignment problems (AP) which can be solved in $O(n^3)$ time [26].

Based on the AP relaxation, several *branch-and-bound* (B&B) algorithms have been developed since 1958. Laporte [26] presents those proposed by Carpaneto and Toth (1980), [27] Balas and Cristophides (1981), [28] Miller and Pekny (1991) [29] among the best available, which provided the optimal solution for randomly generated problems with thousands of nodes (and for some real problems as well) in a reasonable CPU time. Various lower bound procedures were embedded in the B&B algorithms in order to improve their performances, but literature describes that their success depends essentially on the type of the problem to be solved. The symmetric TSPs are better handled by such algorithms [30].

The *branch-and-cut* (B&C) algorithms calculate series of increasing lower and decreasing upper bounds of the optimal solution. The upper bounds are given by heuristic algorithms, while for the lower bounds, the algorithms use a polyhedral cutting-plane procedure over the system of linear inequalities [31]. As a combination between B&B and a cutting procedure, B&C is usually faster than B&B alone. When the cutting-plane procedure does not terminate with an optimal solution, the algorithm uses a tree-search strategy that produces cuts after branching. A B&C algorithm eventually returns the optimal solution when the upper and lower bounds coincide. If this situation is not reached, the quality of the feasible solution can be precisely estimated [32]. The research performed by Jünger et al. [33] extensively describes the branch-and-cut implementation details.

Solvers and other software packages for the TSP

A *solver software* takes an instance of a problem as input, applies one or more methods and provides the result. A more extensive concept is that of the *modeling software*, which provides an environment for formulating, solving and analysing a problem. Modeling software implies at least one solver, but it usually offers several solvers. The following paragraphs provide information related to such products that allow developing research on the TSP. In [34], the reader can find a list of general-purpose software and libraries (released until 2007) for approaching the TSP applications. A survey realized by OR/MS Today Magazine [35] in June 2015 presents a significant list of Linear Programming software packages, together with

the corresponding types (solver/modeling environment/integrated solver and modeling environment), supported platforms, vendors and pricing information.

COIN-OR (Common Optimization INterface for Operations Research) Repository [36] points to a large collection of library of interoperable software tools for building optimization codes, and it also includes several stand-alone packages. It was developed within the COIN-OR (COmputational INfrastructure for Operations Research) Project [37], an initiative promoting the use of interoperable, open-source software for Operations Research. The branch, cut and price library (written in C++) gives researchers the framework for customizing the LP solver according to their needs. COIN-OR also includes an object-oriented Tabu Search framework.

BOB (BOB++) [38] is a general-purpose software library that implements solvers for combinatorial optimization problems on parallel and sequential machines. The methods used are branch-and-bound and divide-and-conquer.

ABACUS (A Branch-And-Cut System) [33, 39] is an open source C++ system providing a framework for the implementation of the B&B and B&C algorithms.

The Concorde TSP Solver package, proposed in 2001, is still one of the fastest TSP exact solvers [40]. It uses the branch-and-cut and the Chained Lin-Kernigan implementations. It is freely available at [41].

Heuristics for the TSP

The heuristics for the TSP can be broadly classified in three classes: *Tour construction procedures*, *Tour improvement procedures* and *Composite procedures*.

The heuristics of the first type gradually build a solution by adding a new vertex at each step. The *Nearest neighbour heuristic* and the *insertion procedures* fall into this class. The insertion procedures use various criteria [1].

The heuristics of the second type use *improvement procedures* that start with a feasible solution and perform various exchanges upon it. The procedures proposed by Lin [42], Lin and Kernigan [43], and Or [44] are known as classical improvement procedures, lying behind the numerous attempts of improvements and hybrid versions.

The *r-opt* algorithm proposed by Lin [42] generalizes the systematic *2-opt* method suggested by Croes [45] in 1958. At a given iteration, *r* arcs (or edges, if the problem is symmetric) are removed from the current tour and all the possible reinsertions are attempted. The best is implemented and the operation is repeated until no further improvement is possible [46]. In 1973, Lin and Kernigan [43] developed the idea by allowing *r* to vary during the search, thus introducing a dynamic *r-opt* heuristic. Numerous heuristics for the TSP implemented this approach and proved to be very efficient. The *Or-opt*, introduced by Or in 1976 [44], attempts to improve the current tour by first moving a chain of three consecutive vertices in a different location, until no further improvement can be obtained. The process is repeated with chains of two consecutive vertices, and then with single vertices.

Although the Lin-Kernigan procedure is recognized as (computationally) effective, sometimes simpler heuristics such as *2-opt*, *3-opt* and *r-opt* are easier to implement and provide good performances. In [47], the authors develop and

compare a number of *Or-opt* variants and (*2-opt*, *Or-opt*) hybrid procedures for the symmetric TSP.

The third class of heuristics includes two-phase construction procedures that combine procedures of the two previous types. The *CCAO* (*Convex hull, Cheapest insertion, Angle selection and Or-opt*) heuristic proposed by Golden and Stewart [48] falls into this class.

Metaheuristics for the TSP

The broad area of metaheuristics that proved to be very efficient for the TSP and its variants includes *Single-solution methods* and *Population-based methods*.

The single-solution techniques center on improving one candidate solution. Simulated Annealing uses a decreasing probability for accepting a worse solution, in order to expand the search and to escape from the local optimum [49]. Tabu Search uses memory structures to avoid considering an already visited solution and also randomly accepts worse solutions [50]. The Variable Neighborhood Search employs a set of neighborhoods and systematically changes them when a local optimum is reached [51].

The population-based methods include classical approaches, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), as well as very novel meta-heuristics. Among these, the Imperialist Competitive Algorithm (ICA), the Artificial Bee Colony (ABC) and the Firefly Algorithm (FA) have already been implemented and applied in various contexts. While ICA [52] simulates the imperialist competition of the country, the nature-inspired methods copy the efficient social behaviour of populations such as ants, fireflies or bees, and different types of communication that these insects exhibit. Ants use pheromones for finding the shortest paths to the food source [53, 54], fireflies use bioluminescent communication [55], and the honey bee swarm divides its tasks between the cooperating groups of employed bees, onlookers and scouts [56]. In all these cases, various principles of communication and collaboration of such simple agents emerge intelligent behaviour at the community level [57–59].

The Bat Algorithm (BA) is a population-based metaheuristic introduced in 2010. The real bats can find their prey and differentiate among different kinds of insects in complete darkness [60]. This characteristic, based on echolocation, has been modeled and led to the BA, firstly proposed by Yang [61] for solving continuous problems. The Fuzzy Logic BA (FLBA) [62] and the Chaotic BA (CBA) [63] are among the versions [64] further developed for the BA, with very interesting applications such as in the study of the dynamic systems. As reviewed in [65], numerous versions of the Binary BA (BBA) were designed for addressing very diverse discrete optimization problems such as selection, planning, and flow shop scheduling. In 2015, an improved version (IBA) was proved [65] to perform with better performances, both for the TSP and for the ATSP.

The TSP Inexact Solvers

Based on the numerous available metaheuristics, the inexact solvers have been developed. The solver LKH [66] is a stochastic local search algorithm, based on the

Lin-Kernighan procedure. This optimization procedure involves swapping sets of k edges to generate feasible, possibly better solutions. It generalizes the 2-opt and 3-opt basic moves by dynamically deciding the value of the parameter k and seeking for better k -opt moves [67]. The LKH was known as the best inexact solver since 2000 and remained so until the development of EAX.

EAX has been recently introduced by Nagata and Kobayashi [68]. It is an evolutionary algorithm that uses 2-opt local search (for the initial population), a Tabu Search procedure (to generate offspring from high-quality parent solutions) and—as a specific feature—an edge assembly crossover procedure (providing very good quality tours by combining two parent tours with a small number of new, short edges). Some experiments [68] ran over commonly studied the Euclidean TSP instances provided, in some cases, with better results than those returned by the LKH.

Encouraging results were published in 2015 by Kotthoff et al. [69]. A series of empirical investigations with LKH and EAX witnessed performance improvements through the algorithm selection techniques [69], showing that per-instance selection between the two solvers can be helpful especially for large the TSP instances.

2.2.3 *Benchmarks for TSP*

In order to assess the performance of the solutions proposed by researchers for the TSP, various benchmarks have been proposed and maintained, starting with the TSPLIB collection initiated in 1990 by Reinelt [70]. Test instances are now available for various geographic problems, for the national TSPs as well as for the collection designed for the industrial applications [71]. Other collections that are the most accessed by researchers are: the Algorithm Selection Benchmark Repository ASlib [72], the benchmark instances for the Traveling Salesman Problem with Time Windows [73]. For our computational study we have used the online geographical database from the Geonames repository [74].

2.3 **Computational Experiment Methodology and Implementation**

The common approach when facing a problem (or a given set of problem instances) is to design and to implement an algorithm for solving it (them) efficiently. The work presented in the following sections uses a specific solver, dedicated to a specific problem, and explores its behavior when solving a set of connected instances. The instance dimension, restrictions and structure can have an important effect on the resources needed by the solving application. Predicting the execution time and/or the optimum solution when solving a specific problem instance can be

very important in operational decisions, for example in case of disasters or military attacks [75, 76].

Discovering difficult instances can be helpful for solvers' designers, who can continuously improve their applications to better treat these reluctant cases. Extracting features that influence the difficultness of an instance can orient the research in optimization through narrowing the investigated solution space and consequently in finding more quickly the optimum solution. Anticipating the solving time for an instance, when knowing the solving time for a close instance, can be a decisive factor in choosing between computerized industrial or financial support systems.

The theoretical concept of re-optimization introduced by Böckenhauer [77] is very close to this empiric investigation. The basic idea of re-optimization is to use the knowledge gathered by solving the previous similar instances. The re-optimization investigations on the TSP consider that two instances are similar if they have the same dimension and their cost functions differ only on one edge. In our work, two instances are similar if one is obtained by adding one node into the other. This experiment is also in line with other early research [78, 79].

Our computational experiment, which is in line with other early research, investigates the behavior of the NEOS server [80] for Concorde when solving a sequence of similar TSP instances. Taking into account that the classical benchmarks earlier mentioned consist of different classes of quasi-independent instances, we constructed a sequence of 31 highly correlated instances.

We started by downloading the data from [81]. This file contains all the world localities with more than 15,000 inhabitants. Each city is specified by the geographic coordinates in decimal format (one real number for the latitude and one real number for the longitude, with positive values for North and East, respectively). This format follows the current ISO recommendation: "For computer data interchange of latitude and longitude, ISO 6709:2008 generally suggests that decimal degrees be used" [82].

When extracting only the European cities and sorting them in a decreasing order by population, we obtained a list with 5978 cities. We decided to derive a sequence where each instance differs from the precedent one by a single node. The instance *europe5000.tsp* has the first 5000 most populous European cities. We added 30 times the next node from the list with 5978 European cities and thus we obtained a sequence of 31 real-world TSP instances. The instances in this collection are specified by their geographic coordinates, since the orthodromic (great-circle) distance between two geo-points on the Earth surface is at the core of the current GIS technologies, which are becoming an essential part of our postindustrial world's digital infrastructure.

Since each instance adds just one node to the previous instance, we expected the following:

- the execution time to slightly differ between two neighbor instances, and
- the optimum tour for the next instance to be likely to simply connect the new node to the optimum tour of the current instance (as in Fig. 2.3a), where the new

Table 2.1 Optimum length (m), solving time (s), type of insertion, the new city and its country for each of the 31 TSP instances

# of nodes	Optimum length (m)	Total solving time (s)	Simple insertion (Yes/No)	Inserted city	Country
5,000	112,648,571	8,743	–	–	–
5,001	112,650,156	9,187	Yes	la Nucia	ES
5,002	112,817,600	6,909	Yes	Mo i Rana	NO
5,003	112,871,915	4,141	No	Forssa	FI
5,004	112,873,153	3,803	Yes	Zuerich (Kreis 11)/ Seebach	CH
5,005	112,903,052	5,893	No	Osuna	ES
5,006	112,911,768	7,912	Yes	Brixham	GB
5,007	112,912,015	7,735	Yes	Amorebieta	ES
5,008	112,912,507	6,746	Yes	Oria	ES
5,009	112,920,189	3,512	No	Reinheim	DE
5,010	112,920,864	5,897	Yes	Kristinehamn	SE
5,011	112,921,034	5,091	Yes	Libiaz	PL
5,012	112,949,261	6,701	Yes	Kukes	AL
5,013	112,949,777	8,158	Yes	Bastia Umbra	IT
5,014	112,952,180	9,377	Yes	Maesteg	GB
5,015	112,977,334	7,454	No	Acqui Terme	IT
5,016	112,983,928	7,917	No	Bunschoten	NL
5,017	112,983,944	8,367	Yes	Bilopillya	UA
5,018	112,990,645	10,905	Yes	Holzwickede	DE
5,019	112,999,351	9,979	Yes	Cercola	IT
5,020	113,008,809	12,251	No	Bohoduksiv	UA
5,021	113,008,928	9,419	Yes	Orsay	FR
5,022	–	–	–	Renens	CH
5,023	113,015,107	11,300	Yes, Yes	Brakel	DE
5,024	–	–	–	Saint-Amand-les-Eaux	FR
5,025	–	–	–	Teo	ES
5,026	113,024,494	10,573	No, Yes, Yes	Zubia	ES
5,027	113,032,048	10,314	Yes	Weesp	NL
5,028	113,061,960	9,961	No	Weissenburg in Bayern	DE
5,029	113,073,099	12,530	No	Palanga	LT
5,030	113,076,593	8,622	Yes	Moita	PT

node Brixham is connected by two red edges, and the blue edge between Paignton and Torquay is deleted.

The results presented in the next section show that both our suppositions are wrong.

2.4 Results and Discussion

In order to test these hypotheses, we called the solver Concorde with default parameters and we recorded for each instance: the execution time, the optimum length, and whether the optimum tour for the next instance differs by only two edges from the optimum tour for the previous one. The results are presented in Table 2.1.

Figure 2.1 shows the optimum tour for the *europa5000.tsp* instance. All the maps from this paper are drawn with the GPSVisualizer [83].

Figure 2.2 presents the 30 new added nodes. They are sampled all over Europe, close to the repartition of the initial 5000 nodes. The countries with the most cities in *europa5000.tsp* are: Germany (882 nodes), Great Britain (610 nodes), France (529 nodes) and Italy (468 nodes). There is no pair of consecutive inserted cities that are close to each other on the map.

Figure 2.3a illustrates the position of the 5006th node (Brixham) into the optimum tour: two red edges are inserted into the optimum tour for the *europa5005.tsp* instance, and the blue edge connecting Paignton with Torquay is deleted. We call

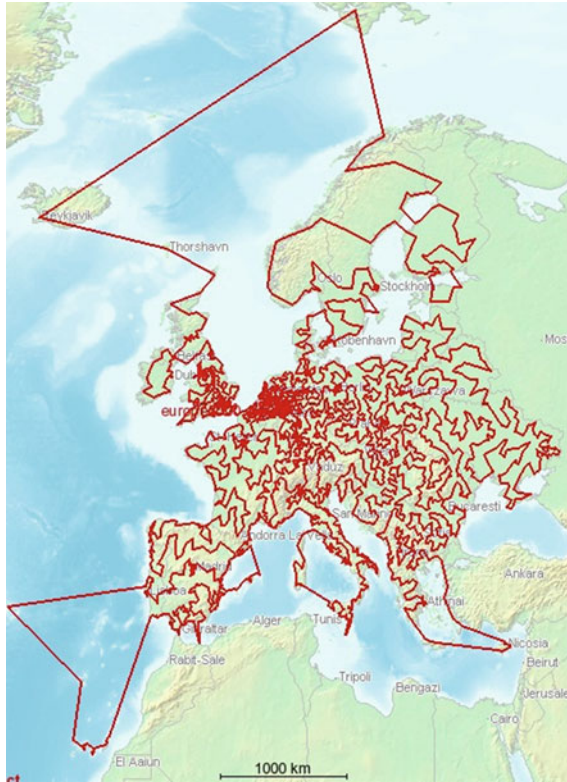


Fig. 2.1 Optimum tour for the *europa5000.tsp* instance

this situation a *simple insertion*. Figure 2.3b shows a more complicated effect of the 5009th node (Reinheim, in the bottom of the image) insertion: the optimum tour for the *europe5008.tsp* instance receives 11 red edges, loses 10 blue edges and becomes the optimum tour for the *europe5009.tsp* instance.

The administrators of the online Concorde solver kill the execution jobs after 4 h, if this is unsuccessful. The first unsolved instance in our experiment was *europe5022.tsp*. The application could not provide the optimum solution in the allocated time. It is interesting that *europe5023.tsp* was solved in 3 h and 9 min, and only two simple insertions into the optimum tour for the *europe5021.tsp* instance were needed (Fig. 2.4).

The next two instances are also difficult. The *europe5026.tsp* instance was solved in less than 4 h (actually in less than 3 h). Its optimum tour has two simple insertions (Fig. 2.5) into the previous, known optimum tour, but it also has a very large tour modification (Fig. 2.6). The optimum tour for *europe5026.tsp* also contains a major path of the same length (red path, central part of Fig. 2.6).

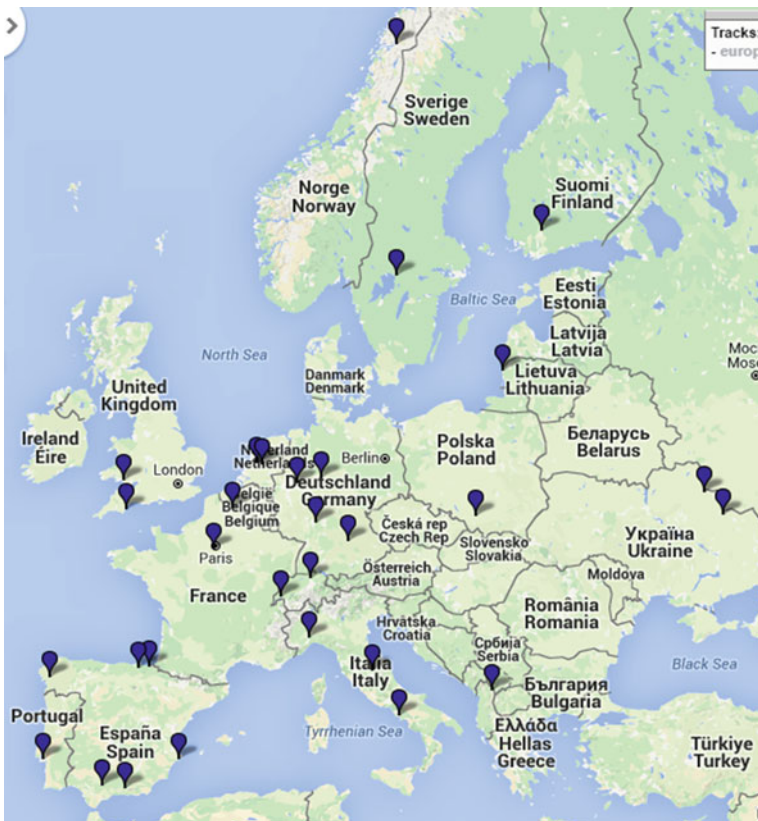


Fig. 2.2 The representation of the 30 nodes sequentially added

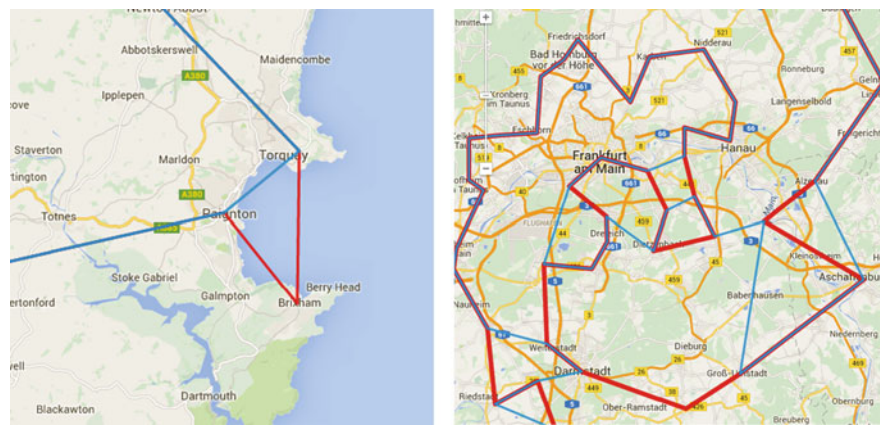


Fig. 2.3 **a** Optimum tour for the europe5006.tsp instance: simple insertion into the europe5005.tsp optimum tour (two new red edges instead of one blue edge). **b** Optimum tour for the europe5009.tsp instance: 11 new red edges replace 10 blue edges from the optimum tour for europe5008.tsp

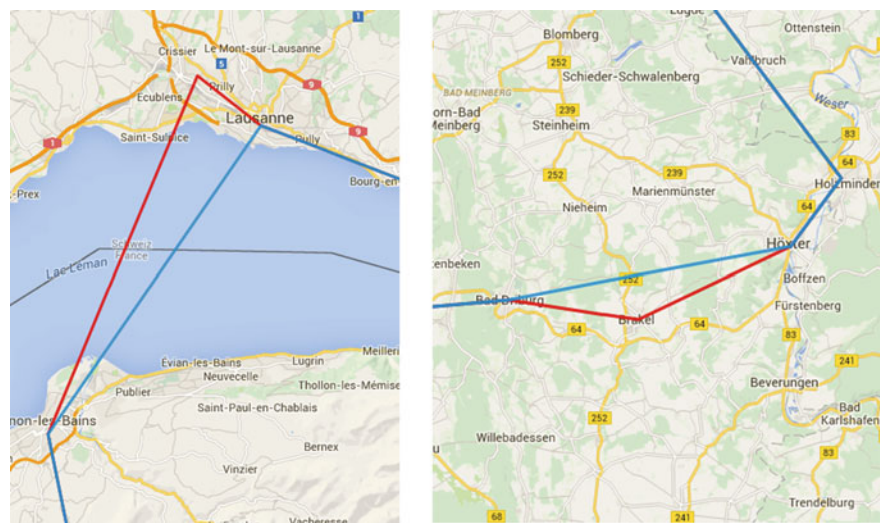


Fig. 2.4 Two simple insertions for the europe5023.tsp optimum tour

The most remarkable change was triggered by Weissenburg in Bayern, which is presented in Fig. 2.7. The new optimum tour covers an estimated area of 1,800,000 square km.

The independent charts representing the optimum tour lengths and the corresponding solving times are displayed in the Figs. 2.8 and 2.9. We computed the Pearson correlation coefficient for the optimum lengths and the solving time. We

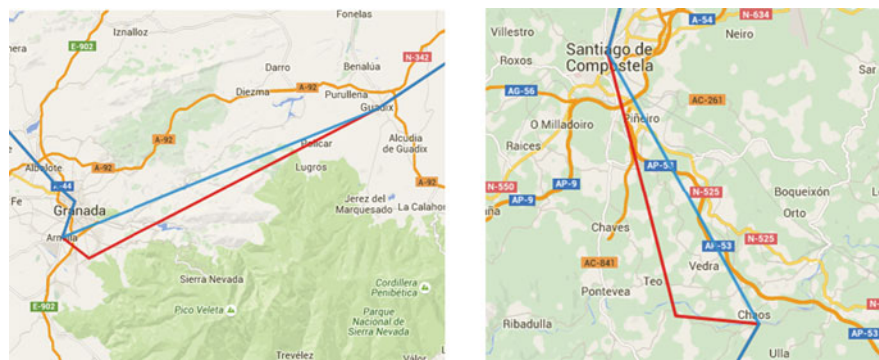


Fig. 2.5 Two simple insertions for the *europe5026.tsp* optimum tour, generated by the two Spanish cities



Fig. 2.6 Large change in the optimum tour for *europe5026.tsp*, generated by the French city (upper left, at the Belgium border), and a new path covering Central Europe

obtained the value $r = -0.041789$, which shows that the two samples do not correlate. This means that if the new city produces a small increase to the tour length, it does not necessarily generate an easier-to-solve instance, and vice versa.

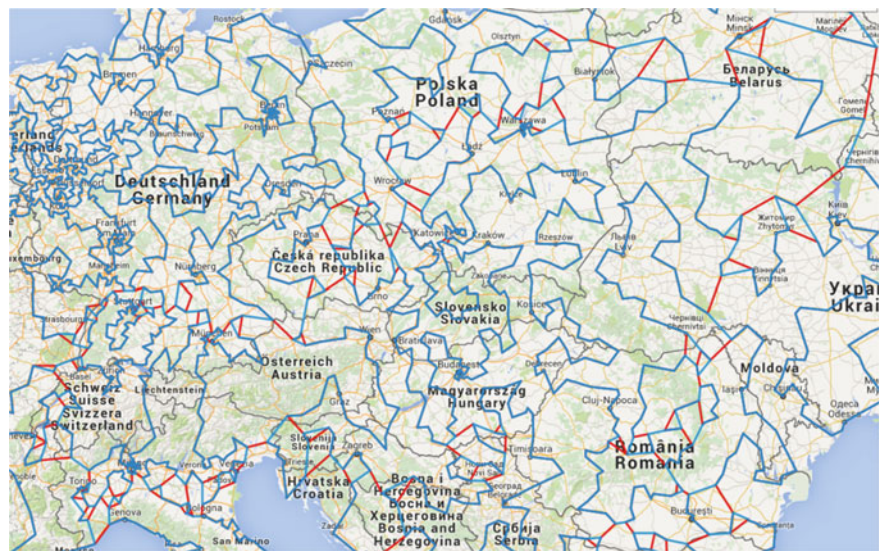


Fig. 2.7 Huge change in the optimum tour for *europe5028.tsp*, generated by the German city (from Ukraine to France, from Poland to Bulgaria, estimated area: 1,800,000 square km)

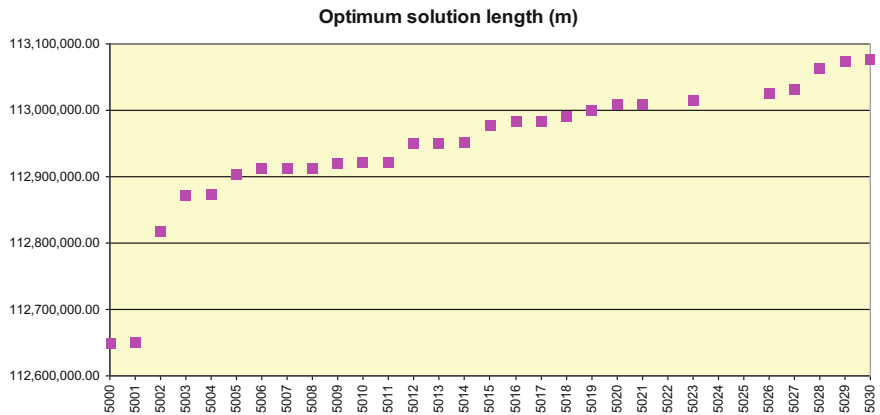


Fig. 2.8 The optimum solution lengths (m) for the TSP instances

For example, *europe5010.tsp* is only 675 m longer than the previous instance and it needed 38 more minutes. The instance *europe5002.tsp* is 167 km longer than *europe5001.tsp*, and it was solved in half an hour less.

According to the Fig. 2.9, we can notice the discontinuity of the solving time. For example, only 9 values lie in a 10 % distance from the previous one. Table 2.1 shows that in 21 cases simple insertions were made. In these cases, there is no pattern manifested in the solving time values. For example, the set *europe5011.tsp*–



Fig. 2.9 The exact solving time (s) for the TSP instances

europe5014.tsp, whose instances have only simple insertions, also has solving times that differ each by more than 1000 s from the previous one. By exclusion, in 30 % of the cases, the effect of the new node was not minimal.

We can conclude that the sequence of the real-world TSP instances we used in our experiment manifest hard-to-predict characteristics of the optimum solutions. Moreover, the solving time was not continuous and the insertion of a new node into the optimum solution was not simple either, as frequent as we expected.

2.5 Conclusions and Future Work

This paper investigates the behavior of an exact TSP solver when it approaches similar instances. We used a set of 31 instances with medium size dimension, representing European localities. Each instance adds a new settlement to the previous one. The results of this empirical study show that no correlation can be highlighted between the lengths of the best solutions, the execution times and the complexity of the transformation of the previous best tour into the current best tour.

Future work has to explore the reasons of this behavior and the structure of the instances that are easier to re-optimize.

The collection of large area, medium dimension, highly correlated TSP instances we defined in this paper can be used to test logistic, emergency management or military decision support systems. Similar real-world, structured collections of instances can be defined by academics. As the current libraries of TSP instances are not correlated, this insight could open new research paths.

Acknowledgments G.C.C. and E.N. acknowledge the support of the project “Bacau and Lugano —Teaching Informatics for a Sustainable Society”, co-financed by a grant from Switzerland through the Swiss Contribution to the enlarged European Union.

References

1. Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.* **40**(6), 1086–1094 (1992)
2. Garey, M.R., Johnson, D.S.: *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, San Francisco (1979)
3. Shafarevich, I.R., Remizov, A.O.: *Linear Algebra and Geometry*. Springer, Berlin (2013)
4. Toth, P., Vigo, D.: An overview of vehicle routing problems. In: *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA (2001)
5. Papadimitrou, C.H., Steiglitz, K.: Some complexity results for the traveling salesman problem. In: *STOC'76 Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, pp. 1–9 (1976)
6. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM* **45**(5), 753–782 (1998)
7. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) *Complexity of Computer Computations, Advances in Computer Research*, pp. 85–103. Plenum Press, (1972)
8. Papadimitrou, C.H.: Euclidean TSP is NP-complete. *Theoret. Comput. Sci.* **4**, 237–244 (1977)
9. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA (2011)
10. Jaillet, P.: *Probabilistic Traveling Salesman Problems*. PhD Thesis, MIT, Cambridge, MA, USA (1985)
11. Henchiri, A., Ballalouna, M., Khansaji, W.: Probabilistic traveling salesman problem: a survey. In: *Position Paper of the 2014 Federated Conference on Computer Science and Information Systems*, pp. 55–60 (2014)
12. Jaillet, P.: A priori solution of a traveling salesman Problem in which a random subset of the customers are visited. *Oper. Res.* **36**, 929–936 (1988)
13. Berman, O., Simchi-Levi, D.: Finding the optimal a priori tour and location of a traveling salesman with nonhomogenous customers. *Transp. Sci.* **22**(2), 148–154 (1988)
14. Laporte, G.: The traveling salesman problem: an overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **59**, 231–247 (1992)
15. Agatz, N., Bouman, P., Schmidt, M.: Optimization approaches for the traveling salesman problem with drone. Technical Report, ERIM report series research in management (2015). <http://repub.eur.nl/pub/78472>
16. Popper, B.: UPS researching delivery drones that could compete with Amazon's Prime Air (2013). <http://www.theverge.com/2013/12/3/5169878/ups-is-researching-its-own-delivery-drones-to-compete-with-amazons>
17. Current, J.R., Schilling, D.A.: The covering salesman problem. *Transp. Sci.* **23**(3), 208–213 (1989)
18. Caric, T., Gold, H. (eds.): *Vehicle Routing Problem*. I-Tech Education and Publishing KG, Vienna (2008)
19. Derigs, U., Pullmann, M., Vogel, U.: Truck and trailer routing—problems, heuristics and computational experience. *Comput. Oper. Res.* **40**(2), 536–546 (2013)
20. Murray, C.C., Chu, A.G.: The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery. *Transp. Res. Part C: Emerg. Technol.* **54**, 86–109 (2015)
21. Hernández-Pérez, H., Salazar-González, J.J.: Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transp. Sci.* **38**(2), 245–255 (2004)
22. Fischetti, M., Lodi, A.: Local branching. *Math. Program.* **98**, 23–47 (2003)
23. Berbeglia, G., Cordeau, J.-F., Laporte, G.: Dynamic pickup and delivery problems. *Eur. J. Oper. Res.* **202**(1), 8–15 (2010)
24. Reinelt, G.: *The Traveling Salesman: Computational Solutions for TSP Applications*. Lecture Notes in Computer Science. Springer, Berlin (1994)

25. Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solutions of a large-scale traveling salesman problem. *Oper. Res.* **2**, 393–410 (1954)
26. Laporte, G., Louveaux, F., Mercure, H.: A priori optimization of the probabilistic traveling salesman problem. *Oper. Res.* **42**(3), 543–549 (1994)
27. Carpaneto, G., Toth, P.: Some new branching and bounding criteria for the asymmetric travelling salesman problem. *Manage. Sci.* **26**, 736–743 (1980)
28. Balas, E., Christofides, N.: A restricted lagrangean approach to the traveling salesman problem. *Math. Program.* **21**, 19–46 (1981)
29. Miller, D.L., Pekny, J.F.: Exact solution of large asymmetric traveling salesman problems. *Science* **251**, 754–761 (1991)
30. Schrijver, A.: *Combinatorial optimization: polyhedra and efficiency*, vol. 1. Springer, Berlin (2003)
31. Padberg, M., Rinaldi, G.: A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* **33**(1), 60–100 (1991)
32. Mitchell, J.E.: *Branch-and-cut algorithms for combinatorial optimization problems*. Handbook of Applied Optimization. Oxford University Press, Oxford (2000)
33. Jünger, M., Reinelt, G., Thienel, S.: Provably good solutions for the traveling salesman problem. *Zeitschrift für Operations Research* **22**, 83–95 (1998)
34. Gutin, G., Punnen, A.P. (eds.): *The Traveling Salesman Problem and Its Variations*. Springer, New York (2007)
35. OR/MS Today magazine, Institute for Operations Research and the Management Sciences: 2015 Linear Programming Software Survey. <http://www.orms-today.org/surveys/LP/LP-survey.html>
36. COIN-OR resources. <http://www.coin-or.org/projects/>, <http://www.coin-or.org/resources.html>
37. COIN-OR Project. <http://www.coin-or.org/>
38. Galea, F., Le Cun, B.: Bob++: a Framework for exact combinatorial optimization methods on parallel machines. In: *Proceedings of the 21st European Conference on Modelling and Simulation* (2007)
39. ABACUS system. <http://www.informatik.uni-koeln.de/abacus/index.html>
40. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: TSP cuts which do not conform to the template paradigm. *Computational Combinatorial Optimization*. Springer, Berlin (2001)
41. Concorde TSP solver. <http://www.math.uwaterloo.ca/tsp/concorde/>
42. Lin, S.: Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* **44**, 2245–2269 (1965)
43. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* **21**, 972–989 (1973)
44. Or, I.: *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*, PhD thesis, North-Western University, Evanston, IL (1976)
45. Croes, G.A.: A method for solving large scale symmetric traveling salesman problems to optimality. *Oper. Res.* **6**, 791–812 (1958)
46. Rosenkrantz, D.J., Stearns, R.E., Philip, I., Lewis, M.: An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.* **6**(3), 563–581 (1977)
47. Babin, G., Deneault, S., Laporte, G.: Improvements of the Or-opt Heuristic for the Traveling Salesman Problem. GERARD—Group for Research in Decision Analysis. Montreal, Quebec, Canada (2005). <https://blogue.hec.ca/permanent/babin/pub/Babi05a.pdf>
48. Golden, B.L., Stewart, Jr.W.R.: Empirical analysis of heuristics. In: Hawler, E.L., Lenstra, J. K., Rinnouy Kan, A.H.G., Shmoys, D.B. (eds.) *The Traveling Salesman Problem*, pp. 207–249. Wiley, New York (1985)
49. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
50. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic Publishers (1997)
51. Hansen, P., Mladenovic, N., Perez, J.A.M.: Variable neighbourhood search: methods and applications. *Ann. Oper. Res.* **175**, 367–407 (2010)

52. Ardalan, Z., Karimi, S., Poursabzi, O., Naderi, B.: A novel imperialist competitive algorithm for generalized traveling salesman problems. *Appl. Soft Comput.* **26**, 546–555 (2015)
53. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
54. Crişan, G.C., Pinteau, C.M., Pop, P.: On the resilience of an ant-based system in fuzzy environments. an empirical study. In: *Proceedings of the 2014 IEEE International Conference on Fuzzy Systems*, Beijing, China, pp. 2588–2593 (2014)
55. Jati, G.K., Suyanto, S.: Evolutionary discrete firefly algorithm for traveling salesman problem, *ICAIS 2011. Lecture Notes in Artificial Intelligence (LNAI 6943)*, pp. 393–403 (2011)
56. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput.* **8**(1), 687–697 (2008)
57. Iantovics, B., Chira, C., Dumitrescu, D.: *Principles of Intelligent Agents*. Casa Cărţii de Ştiinţă, Cluj-Napoca (2007)
58. Nechita, E., Muraru C.V., Talmaciu M.: Mechanisms in social insect societies and their use in optimization. a case study for trail laying behavior. In: *Proceedings of the 1st International Conference Bio-Inspired Computational Methods Used for Solving Difficult Problems—Development of Intelligent and Complex Systems BICS'2008*, Târgu Mureş, AIP Conference Proceedings, Melville, New York (2009)
59. Pinteau, C.M.: *Advances in Bio-inspired Computing for Combinatorial Optimization Problems*. Springer, Berlin (2014)
60. Brigham, R.M., Kalko, K.V., Jones, G., Parsons, S., Limpens, H.J.G.A (Eds.): *Bat echolocation research: tools, techniques and analysis*, Austin, Texas (2002)
61. Yang, X.S.: A new meta-heuristic bat-inspired algorithm. In: Gonzales, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp. 65–74. Springer, Berlin (2010)
62. Khan, K., Nikov, A., Sahai, A.: A fuzzy bat clustering method for ergonomic screening of office workplaces. In: *Third International Conference on Software, Services and Semantic Technologies*, pp. 59–66. Springer (2011)
63. Lin, J.H., Chou, C.W., Yang, C.H., Tsai, H.L.: A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. *J. Comput. Inf. Technol.* **2**(2), 56–63 (2012)
64. Yang, X.S., He, X.: Bat algorithm: literature review and applications. *Int. J. Bio-Inspired Comput.* **5**, 141–149 (2013)
65. Osaba, E., Yang, X.S., Diaz, F., Lopez-Garcia, P., Carballedo, R.: An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems, *engineering applications of artificial intelligence* (2015, in press)
66. Helsgaun, K.: General k-opt submoves for the Lin-Kernighan TSP heuristic. *Math. Program. Comput.* **1**(2–3), 119–163 (2009)
67. Helsgaun, K.: *Solving the Bottleneck Traveling Salesman Problem Using the Lin-Kernighan-Helsgaun Algorithm*. Technical Report, Computer Science, Roskilde University (2014)
68. Nagata, Y., Kobayashi, S.: A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS J. Comput.* **25**(2), 346–363 (2013)
69. Kotthoff, L., Kerschke, P., Hoos, H., Trautmann, H.: Improving the state of the art in inexact TSP solving using per-instance algorithm selection. *Lecture Notes in Computer Science*, vol. 8994, pp. 202–217. Springer (2015)
70. Library of various sample TSP and TSP-related instances. <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>
71. TSPLIB. <http://www.math.uwaterloo.ca/tsp/index.htm>
72. Algorithm Selection Library ASlib. <http://www.coseal.net/aslib/>
73. Benchmark instances for the Traveling Salesman Problem with Time Windows. <http://lopez-ibanez.eu/tspw-instances>
74. Geonames repository. <http://www.geonames.org>
75. Crişan, G.C., Pinteau, C.M., Chira, C.: Risk assessment for incoherent data. *Environ. Eng. Manag. J.* **11**(12), 2169–2174 (2012)

76. Nechita, E., Muraru, C.V., Talmaciu, M.: A Bayesian approach for the assessment of risk probability. *Case Study Dig. Risk Probab. Environ. Eng. Manag. J.* **11**(12), 2249–2256 (2012)
77. Böckenhauer, H.J., Hromkovič, J., Mömke, T., Widmayer, P.: On the Hardness of Reoptimization, *SOFSEM 2008. LNCS*, vol. 4910, pp. 50–65. Springer, Heidelberg (2008)
78. Papadimitriou, C.H., Steiglitz, K.: Some examples of difficult traveling salesman problems. *Oper. Res.* **26**(3), 434–443 (1978)
79. Ahammed, F., Moscato, P.: Evolving L-systems as an intelligent design approach to find classes of difficult-to-solve traveling salesman problem instances. In: *Applications of Evolutionary Computation EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC*, Torino, Italy, April 27–29, 2011, *Proceedings, Part I*, pp. 1–11. Springer, Berlin (2011)
80. NEOS server. <http://www.neos-server.org/neos/>
81. World cities with 15,000 people or more. <http://download.geonames.org/export/dump/>
82. ISO 6709:2008, Standard representation of geographic point location by coordinates. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39242
83. GPSVisualizer. <http://www.gpsvisualizer.com>

Advances in Combining Intelligent Methods

Postproceedings of the 5th International Workshop
CIMA-2015, Vietri sul Mare, Italy, November 2015 (at
ICTAI 2015)

Hatzilygeroudis, I.; Palade, V.; Prentzas, J. (Eds.)

2017, XI, 147 p. 40 illus., Hardcover

ISBN: 978-3-319-46199-1