Alexander K. Hartmann and Martin Weigt

Phase Transitions in Combinatorial Optimization Problems

Basics, Algorithms and Statistical Mechanics

WILEY-VCH Verlag GmbH & Co. KGaA

Alexander K. Hartmann
Martin Weigt

Phase Transitions in Combinatorial Optimization Problems

Basics, Algorithms and Statistical Mechanics

Related Titles

Hartmann, A. K. / Rieger, H. (eds.)
New Optimization Algorithms in Physics
2004. XII, 300 pages with 113 figures. Hardcover.
ISBN 3-527-40406-6

Gen, M. / Cheng, R.
Genetic Algorithms and Engineering Optimization
Series: Wiley Series in Engineering Design and Automation
2000. XVI, 496 pages. Hardcover.
ISBN 0-471-31531-1

Wolsey, L. A. / Nemhauser, G. L.
Integer and Combinatorial Optimization
Series: Wiley-Interscience Series in Discrete Mathematics and Optimization
1999. XVI, 764 pages. Softcover.
ISBN 0-471-35943-2

Chong, Edwin K. P. / Zak, Stanislaw H.
An Introduction to Optimization
Juli 2001, 496 pages. Hardcover.
ISBN 0-471-39126-3

Deb, Kalyanmoy
Multi-Objective Optimization Using Evolutionary Algorithms
Juni 2001, 518 pages. Hardcover.
ISBN 0-471-87339-X

Fletcher, Roger
Practical Methods of Optimization
Juni 2000, 450 pages. Softcover.
ISBN 0-471-49463-1

Alexander K. Hartmann and Martin Weigt

Phase Transitions in Combinatorial Optimization Problems

Basics, Algorithms and Statistical Mechanics

WILEY-VCH Verlag GmbH & Co. KGaA

The Authors of this Book

Dr. Alexander K. Hartmann
University of Goettingen,
Institute for Theoretical Physics
hartmann@theorie.physik.uni-goettingen.de

Dr. Martin Weigt
I.S.I. Institute for Scientific
Interchange Foundation, Turin
weigt@isiosf.isi.it

Library of Congress Card No.: applied for
British Library Cataloging-in-Publication Data:
A catalogue record for this book is available from the British Library.
Bibliographic information published by Die Deutsche Bibliothek.
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at http://dnb.ddb.de.

Printing Strauss GmbH, Mörlenbach

Binding Litges & Dopf Buchbinderei GmbH, Heppenheim

Printed in the Federal Republic of Germany

Printed on acid-free paper

ISBN-13: 978-3-527-40473-5
ISBN-10: 3-527-40473-2

Contents

Preface

This text book provides an introduction to the young and growing research area of statistical mechanics of combinatorial optimization problems. This cross-disciplinary field is positioned at the borderline between physics, computer science, and mathematics. Whereas most optimization problems and the majority of available computer algorithms used for their solution depend on computer science and mathematics, the main methods presented in this book are related to statistical physics.

Combinatorial optimization problems originate in real-world applications. The head of the local post office wants to distribute a lot of parcels in a city using the minimum of resources such as the number of lorries, the amount of fuel and the number of personnel. The director of a school wants to set up a schedule so that no teacher has classes in parallel while at the same time minimizing their idle times. The participant of a scientific meeting wants to find a fast train connection to the location of a meeting, which does not require too many changes of trains. Optimization problems also occur very frequently in science, such as determining the ground states of physical systems, identifying the native states of proteins, or looking for similarities in ancient languages. Computer scientist have identified certain prototypes of problems behind these real-world and science applications, like finding the shortest paths, the shortest round trips, or satisfying the conditions of Boolean formulas.

For several decades, development in this field took place first of all in applied computer science. The basic task was, and still is, to find algorithms which solve these problems as quickly as possible. Also theoretical computer science, within *algorithmic complexity theory*, considered these problems, in order to classify them according to their "difficulty". A major breakthrough came in 1971, when Cook proved the *NP-completeness* of the *satisfiability problem* (SAT), i. e., he was able to show that SAT is equivalent to all problems from the class NP, which contains the most interesting, i. e., not easily solvable, problems. This breakthrough allowed for a better classification of the problems and has fertilized the field. Since then, literally thousands of problems have been identified as being NP-complete as well. Nevertheless, the notion of NP-completeness is related to the *worst-case* running time of an algorithm. For all NP-complete problems, all known algorithms exhibit a worst-case running time which explodes exponentially with the problem size. However, to study real-world applications, one would also like to develop a notion of *typical-case* complexity. For this purpose, computer scientists started to study the behavior of algorithms defined on suitable parametrized ensembles of random systems. Interestingly, threshold phenomena were found, which were related to a drastic change in the structure of optimal solutions as well as in the typical running time of algorithms. These phenomena show surprising similarities to *phase transitions* in physical systems.

Here statistical physics comes into play, because the consideration of the typical behavior for systems of many particles is the fundamental problem in this field. Many tools have been developed in statistical physics, which allow the study of phase transitions and glassy behavior, which are two basic phenomena occurring in the study of the typical behavior of combinatorial optimization problems. The transfer of knowledge from statistical physics to computer science started in the mid 1980s, when the *simulated annealing method* was invented, which is a fast yet simple algorithm allowing one to find heuristically, very good approximate solutions of combinatorial optimization problems. The idea of this approach is to map the cost function of the optimization problem on the energy function of an equivalent physical system. On the basis of this type of mapping, from the mid 1990s on, concepts and methods from statistical mechanics were applied to study the phase transitions occurring in optimization problems. Since then, many new results have been found, which have not been accessible before by using solely mathematical tools. Also other new algorithms have been developed on the foundation of a physics perspective, e. g., the *survey propagation approach* which allows the treatment of much larger problems, e. g., for SAT, than before. Given the large number of NP-complete problems, the growing range of applications of combinatorial optimization and also the growth of computer power, much progress in this direction can be expected in forthcoming years.

In this book, we introduce concepts and methods for the statistical mechanics of disordered systems. We explain models exhibiting *quenched disorder* such as spin glasses and hard-core gases. We introduce the most important analytical techniques in this field, namely the *replica approach* and the *cavity method*. We also present stochastic algorithms like the Monte-Carlo method or the survey propagation technique, both of which are based on a statistical mechanics viewpoint. In this way the book serves also as an introduction to the statistical mechanics of disordered systems, but with the special point of view of combinatorial optimization.

As in many interdisciplinary contexts, culturally different approaches and languages are present in the different scientific communities. Whereas, e. g., mathematical research work considers mathematical rigor as one of the fundamental building blocks, physical research is usually much more "result oriented". To understand complicated natural phenomena, it is frequently necessary to work with approximations, or to use assumptions which are not immediately justified mathematically.

One aim of this book is helping to bridge the resulting language gap. The book is intended to be accessible for readers with different backgrounds: physicists, who want to enter into this new interdisciplinary area, mathematicians and theoretical computer scientists, who want to understand the methods used by physicists, or applied computer scientists, who want to exploit the insight gained using physics tools to develop new and more efficient algorithms. The level of the book is targeted to graduate students of the different fields, who want to enlarge their horizon beyond the specific borders of their subject of study.

To achieve this, we include various introductory chapters on algorithms, graph theory, complexity theory, and on statistical mechanics. These chapters formulate the minimal basic knowledge for the main part of the book. Readers educated in these fields may want to skip some parts here.

In the central part of the book, we concentrate, for pedagogical reasons and internal coherence, on one specific model, the *vertex cover problem*. We have selected this problem because, on one hand, it shows the main phenomena discussed in the field. On the other hand, the analytical approaches are relatively easy to access compared, e. g., with the most famous combinatorial problem, the satisfiability problem. The details of the latter model are discussed in a specific chapter towards the end of the book. We think that this pedagogical approach enables the reader to understand more easily the vast original literature.

After having discussed in detail the application of statistical mechanics tools to computer-science problems, we also include a chapter discussing the information transfer in the opposite direction, which is historically more strongly developed. Many physical problems can be understood as combinatorial optimization problems. Using various examples, we discuss how computer-science algorithms can be applied to better understand the physical behavior of such systems.

In preparing this book we benefited greatly from many collaborations and discussions with many of our colleagues. We would like to thank Mikko Alava, Simon Alder, Carlo Amoruso, Timo Aspelmeier, Alain Barrat, Wolfgang Barthel, Jürgen Bendisch, Giulio Biroli, Stefan Boettcher, Alfredo Braunstein, Alan Bray, Kurt Borderix, Bernd Burghardt, Ian Campbell, Adam Carter, Loredana Correale, Rodolfo Cuerno, Eytan Domany, Phil Duxbury, Andreas Engel, Martin Feix, Silvio Franz, Ulrich Geyer, Dieter Heermann, Guy Hed, Olaf Herbst, Heinz Horner, Jérôme Houdayer, Michael Jünger, Helmut Katzgraber, Sigismund Kobe, Matthias Koelbel, Werner Krauth, Reiner Kree, Florent Krzakala, Michele Leone, Klaus-Peter Lieb, Frauke Liers, Andreas Linke, Olivier Martin, Alan Middleton, Remi Monasson, Michael Moore, Alejandro Morales, Juan Moreno, Roberto Mulet, Javier Muñoz-García, Uli Nowak, Matthias Otto, Andrea Pagnani, Matteo Palassini, Gerhard Reinelt, Alberto Rosso, Federico Ricci-Tersenghi, Heiko Rieger, Guilhem Semerjian, Eira Seppälä, Dietrich Stauffer, Simon Trebst, Matthias Troyer, Klaus Usadel, Alexei Vazquez, Emmanuel Yewande, Peter Young, Riccardo Zecchina and Annette Zippelius.

This book was prepared at the University of Göttingen and the Institute for Scientific Interchange (ISI), Turin. We would like to acknowledge financial support from the Deutsche Forschungsgemeinschaft (DFG), the VolkswagenStiftung and the European Science Foundation (ESF).

Göttingen and Turin, January 2005

Alexander K. Hartmann and Martin Weigt

1 Introduction

Optimization problems appear in many situations in our daily life. If you, e. g., connect to the web server of a train company or travel agency, you can search connections between far-distant places for optimal travel times, ticket prices or number of changes, etc. In a more general economic context, such problems appear whenever a process has to be arranged in such a way that resources, money, or time are saved. Also in science optimization problems are of central interest, e. g., in physics for understanding the low-temperature behavior of model systems, or in biology for extracting information out of a huge amount of experimental data.

Probably you first encountered optimization in school: In mathematics courses one-dimensional functions over the space of real variables are analyzed, including the determination of minima and maxima. Such problems are in general easily solvable – at least

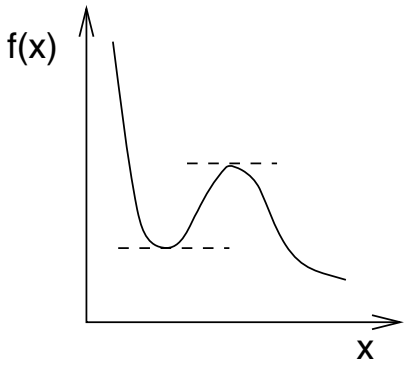

Figure 1.1: Extremal points of a function defined over a one-dimensional continuous space.

on a computer. In this book we are going to discuss so-called combinatorial optimization problems. These are in practice much harder to solve. First, the function to be minimized is in general high-dimensional; second, the variables are of discrete nature, and methods of continuous analysis such as taking derivatives do not work.

In this introductory chapter, we will present two examples which give you a flavor of combinatorial optimization. Furthermore, we will explain why optimization problems are interesting for physicists and how statistical mechanics can contribute to the understanding and solution of them. Finally, we recommend and discuss some basic textbooks related to the field.

1.1 Two examples of combinatorial optimization

Let us start by briefly formalizing the above explanation of combinatorial optimization problems. In general they are defined over some high-dimensional space, we will consider multi-component variables $\underline{\sigma} = (\sigma_1, \ldots, \sigma_n) \in X^n$ with $n \gg 1$. In addition, the problem is characterized by some *cost function* $H : X^n \to \mathbb{R}$ which assigns some cost to each $\underline{\sigma}$. This cost has to be minimized, leading to the following definition:

Definition: *minimization problem*

Given an n-dimensional space X^n and a cost function $H : X^n \to \mathbb{R}$.
The *minimization problem* reads as follows:

$$\textit{Find } \underline{\sigma}^{(0)} \in X^n \textit{, such that } H(\underline{\sigma}) \geq H(\underline{\sigma}^{(0)}) \textit{ for all } \underline{\sigma} \in X^n$$

The vector $\underline{\sigma}^{(0)}$ is called a *solution* of the minimization problem.

Finding a maximum is as easy or as hard as finding a minimum, since $\max H = -\min(-H)$. In addition we speak of *combinatorial* problems if the components of $\underline{\sigma}$ are of *discrete* nature, frequently used examples are $X = \{-1, 1\}$ (Ising spins), $X = \{true, false\}$ (Boolean variables) or $X = \mathbb{Z}$ (integer numbers).

Combinatorial optimization problems can be additionally complicated by the presence of *constraints*. The constraints reduce the size of the search space. At first glance this seems to facilitate the search for an optimal solution. The opposite is, however, frequently the case: Many optimization problems which can be solved efficiently on a computer without constraints, become extremely computer-time consuming if constraints are added.

To illustrate these rather abstract concepts, we will present two examples. The first one is a classical optimization problem in computer science, originating in economics.

Example: Traveling Salesman Problem (TSP)

Consider n towns distributed in a plane, numbered by $1, \ldots, n$. A salesman wants to visit all these towns, and at the end of his travels he wants to come back to his home town. He is confronted with the following minimization task: He has to find the shortest round-tour visiting every town exactly once. The problem is thus described by

$$\begin{aligned} X &= \{1, 2, \ldots, n\} \\ H(\underline{\sigma}) &= \sum_{i=1}^{n} d(\sigma_i, \sigma_{i+1}) \end{aligned} \tag{1.1}$$

where $d(\sigma_i, \sigma_j)$ is the distance between the two towns σ_i and σ_j, and $\sigma_{n+1} \equiv \sigma_1$ are identified with each other. The constraint that every town is visited once and

only once can be realized by constraining the vector $\underline{\sigma}$ to be a permutation of the sequence $(1, 2, \ldots, n)$.

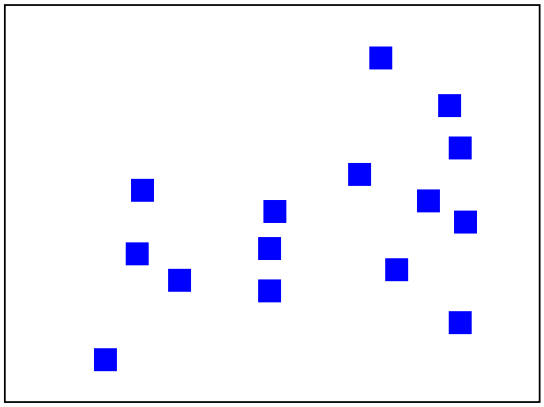

Figure 1.2: 15 cities in a plane.

As an example, 15 towns in a plane are given in Fig. 1.2; and one can try to find the shortest tour. For the general TSP the cities are not necessarily placed in a plane, but an arbitrary distance matrix d is given. □

The TSP is a so-called *hard* optimization problem. In this context, hardness is measured by the time a computer needs to solve this problem numerically, and a problem is considered to be hard if this time grows exponentially (or even faster) with the number n of components of the variable $\underline{\sigma}$. In TSP, one out of $(n-1)!$ round-tours has to be selected, and so far no good general selection criterion exists. Note that TSP has attracted not only computer scientists, but also physicists [1–6].

For an arbitrary algorithm, to describe the dependence between a suitably chosen measure n of the problem size and the running time T, the $\mathcal{O}$ notation is used.

Definition: *$\mathcal{O}$ notation*

Let $T, g : \mathbb{N} \to \mathbb{R}$ be two real-valued functions.
We write $T(n) = \mathcal{O}(g(n))$, iff there exists a positive number $c > 0$, such that $T(n) \leq cg(n)$ is valid for all $n > 0$. We say, $T(n)$ is *of order* at most $g(n)$.

Since constants are ignored when using the $\mathcal{O}$ notation, one speaks of the *asymptotic* running time or *time complexity*. In theoretical computer since, usually on states an upper bound over all possible inputs of size n, i. e., the *worst-case running time*. In this book, we will also study *typical* running times regarding a given ensemble of instances, see Sec. 1.2.

In Table 1.1, orders of running times, which occur typically in the context of algorithms, are presented, accompanied by the resulting values for problem sizes 10, 100, and 1000.

Table 1.1: Growth of functions as a function of input size n.

$T(n)$	$T(10)$	$T(100)$	$T(1000)$
n	10	100	1000
$n \log n$	10	200	3000
n^2	10^2	10^4	10^6
n^3	10^3	10^6	10^9
$n^{\log n}$	10	10^4	10^9
2^n	1024	1.3×10^{30}	1.1×10^{301}
$n!$	3.6×10^6	10^{158}	4×10^{2567}

Usually one considers problems as *easy*, if the running time is bounded by a polynomial, all others are considered as *hard*. The reason can be understood from the table: Even if the polynomial functions may take higher values for small n, asymptotically non-polynomial functions diverge much faster. Let us consider, e. g., the relative performance of two computers, one being twice as fast as the other one. In a linear-time problem, the faster computer is able to solve a problem which is twice as large as the problem solvable in the same time on the slower computer. If the running time grows, however, as 2^n, the faster computer is just able to go from n to $n + 1$ compared with the slower one. We see that for such hard problems, the utility of higher-speed computers is very limited – a substantial increase in the size of solvable problems can only be achieved via the use of better algorithms.

Often it is not immediately obvious whether or not a problem is easy. While finding the shortest round tour in the TSP is hard, finding the shortest path between two given towns (possibly through other cities) is easy, as we will see in Sec. 11.4.

Also in physics, many problems either are, or can be translated into, optimization problems. Examples are the determination of ground states of magnetic systems, the calculation of the structure of a folded protein, the analysis of data, or the study of flux lines in superconductors. This will be illustrated using a classical model in statistical physics.

Example: Ising spin glasses

Spin glasses [7, 8] are amorphous magnetic materials. The atoms in the solid carry microscopically small magnetic moments, which interact via couplings, some are ferromagnetic while others are antiferromagnetic. Due to the amorphous nature of the material, these couplings are disordered, i. e., they do not have any periodic structure. Spin glasses show an interesting frozen low-temperature phase which is, despite ongoing research over more than three decades, still poorly understood. Spin glasses can be modeled in the following way:

The magnetic moments are described by *Ising spins* σ_i which, due to anisotropies of the material, can take only two orientations called *up* and *down*, mathematically formalized by $\sigma_i = \pm 1$. In the simplest model one assumes that these spins are

placed on the sites of a regular lattice and that a spin interacts only with its nearest neighbors. The model is thus described by

$$\begin{aligned} X &= \{-1, 1\} \\ H(\underline{\sigma}) &= -\sum_{<i,j>} J_{ij}\sigma_i\sigma_j \end{aligned} \tag{1.2}$$

where J_{ij} denotes the interaction strength between the spins on sites i and j, and the sum runs over all pairs $\langle i, j\rangle$ of nearest neighbors. The function H measures the total energy of the system, and is called the *Hamiltonian*. Note that the interaction parameters are fixed, they are also called *quenched* variables, whereas the Ising spins are subject to thermal fluctuations and my change their values.

In a *ferromagnet* it is energetically favorable for any two neighboring spins to assume equal orientations $\sigma_i = \sigma_j$, i. e., all J_{ij} are positive, and parallel spins lead to a lower contribution to the total energy than antiparallel ones. So the system tends to be globally oriented in the same way. On the other hand, thermal noise causes spins to fluctuate randomly. At low temperatures T this thermal noise is small, and energetic contributions dominate over random spin fluctuations. The system becomes globally *ordered*. For temperatures higher than some critical temperature T_c, this long-range order becomes destroyed; a *phase transition* occurs at T_c. In Chap. 5, this phenomenon will be discussed in more detail in the context of a short introduction to statistical mechanics.

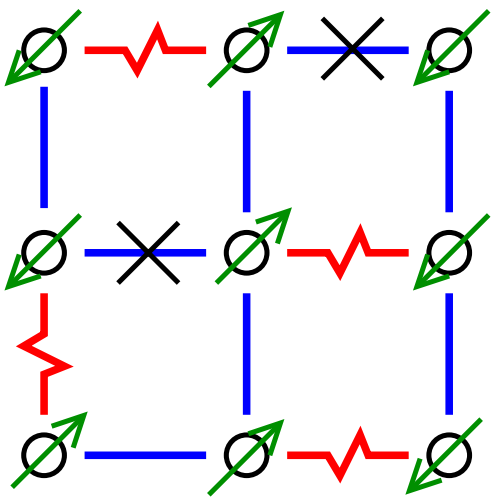

Figure 1.3: Two-dimensional spin glass. Solid lines represent ferromagnetic interactions, while jagged lines correspond to antiferromagnetic interactions. The small arrows represent the spins, adjusted to a ground-state configuration. For all except two interactions (marked by the crosses) the spins are oriented relative to each other in an energetically favorable way. It is not possible to find a state with lower energy.

At temperature $T = 0$ thermal fluctuations are completely absent, and the energy $H(\underline{\sigma})$ of the system assumes its global minimum. The corresponding spin configuration is called a *ground state*. The properties of these configurations are of great interest in physics, they serve as a basis for understanding the low-temperature behavior of physical systems. In the case of a ferromagnet these ground states are, as discussed above, extremely simple; all spins have the same orientation.

This becomes more complicated for *spin glasses* because ferromagnetic interactions ($J_{ij} > 0$) coexist with *antiferromagnetic* ones ($J_{ij} < 0$). Two spins connected by such an antiferromagnetic coupling prefer energetically to have opposite orientations. As already noted, spin glasses are amorphous materials, therefore ferromagnetic and antiferromagnetic interactions are distributed randomly on the bonds of the lattice. Consequently, it is not obvious how the ground state configurations of $H(\underline{\sigma})$ appear, and determining the minimum energy becomes a non-trivial minimization problem.

Figure 1.3 shows a small two-dimensional spin glass and one of its ground states. For this type of system usually many different ground states are feasible for each realization of the disorder. One says, the ground state is *degenerate*. Algorithms for calculating degenerate spin-glass ground states are explained in Chap. 11. As we will explain there, this calculation is easy (polynomial running time) for two-dimensional systems, but it becomes hard (exponential running time) in three dimensions. □

1.2 Why study combinatorial optimization using statistical physics?

The interdisciplinary exchange between computer science and statistical physics goes in both directions:

On one hand, many problems in daily life and in many scientific disciplines can be formulated as optimization problems. Hence, progress in the theory of combinatorial optimization, and in particular the development of new and more efficient algorithms influences many fields of science, technology and economy. In this obvious sense, computer science contributes to the understanding of physical systems. We will give a short introduction on the application of optimization algorithms for physics problems in Chap. 11.

On the other hand, physics can also help to shed light onto some basic, yet unsolved, questions in computer science. In statistical mechanics, many analytical and numerical methods have been developed in order to understand the macroscopic thermodynamic behavior of models starting from some microscopic description of a system via its Hamiltonian. These methods can be reinterpreted from the point of view of optimization theory. Problems, which are originally formulated as purely combinatorial tasks, can be equivalently rewritten as physical

models, by identifying the cost function with a Hamiltonian. Applying statistical mechanics tools at a temperature close to $T = 0$, may thus unveil many properties of the original problem and its cost-function minima; this will be the main focus of this book. In this way we will obtain insight into the intrinsic reasons for the hardness of these problems. For example, we will see that the hardness of many optimization problem is closely related to the *glassiness* of the corresponding physical system. Using this alternative physical approach, many interesting results have already been obtained, which have not been found before by applying traditional methods from mathematics and computer science.

Furthermore, traditional computer science defines the hardness according to a *worst-case* scenario. People dealing with practical applications are, however, more interested in *typical* instances of an optimization task rather than looking for the hardest possible instances. For this reason, suitably parametrized random ensembles of instances of problems have been introduced over recent years. In this context, it was observed that in some regions of the ensemble space instances are *typically* easy to solve, i. e., in a polynomially increasing running time, while in other regions instances are found to be typically hard. This change in behavior resembles the phase transitions observed in physical systems, like spin glasses or other disordered materials. Once more it is tempting to exploit the long experience of statistical physics with phase transitions in order to understand this behavior.

Example: Phase transitions in the TSP

As an example, we consider again the TSP. We study the random ensemble, where a plane of area size $A = L_x \times L_y$ is given. Each random instance consists of n cities, which are randomly placed in the plane, with $(x_i, y_i) \in [0, L_x] \times [0, L_y]$ denoting the position of city i. All positions are equally probable. The distances between pairs of cities are just Euclidean distances, i. e., $d(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

For each random instance of the problem, we ask the question:

Is the shortest round trip through all cities shorter than a given length l?

To answer this question for a given instance, one can use a *branch-and-bound* algorithm. For an introduction to this type of algorithm, see Sec. 6.3. Here, the algorithm basically enumerates all possible permutations of n cities stepwise, by selecting one city after the other. Some efficient heuristic is used, which determines the order in which the cities are tried, e. g., one starts the construction of the round trip, by first taking the two cities which are closest to each other, for details see Ref. [9]. The algorithm furthermore maintains a lower bound $l_{\min}$ of the tour length, determined by the cities selected so far. If $l_{\min} > l$, then the algorithm does not have to pursue permutations containing the same order of the cities selected so far, and it can search in other regions of the permutation space. On the other hand, if a full round trip with $H(\underline{\sigma}) < l$ has been found, one knows that the answer to the question above is "yes", and the algorithm can stop.

In Fig. 1.4, the probability p that a tour of length smaller than l exists, is shown [9] as a function of the rescaled length $\Phi = l/\sqrt{nA}$. One observes that there is a strong

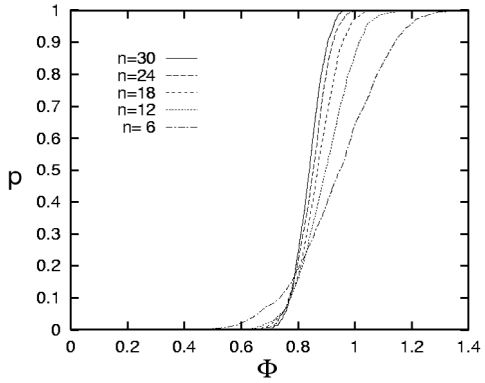

Figure 1.4: Probability p that the minimum tour of n cities in plane of area A is shorter than $\Phi = l/\sqrt{nA}$, for different number n of cities. Reprinted from Ref. [9] with permission from Elsevier.

increase in p when increasing l and that the curves for different sizes cross close to $\Phi = 0.78$. Hence, there seems to be a fundamental change in the nature of the question close to $\Phi = 0.78$, which resembles a *phase transition* in physical systems, see Sec. 5.1.

One can show [9] that all probability curves for different sizes n fall on top of each other, when p is plotted as a function of $r = (l/\sqrt{nA} - 0.78)n^{2/3}$. In this sense, the problem exhibits a certain type of universality, independent of the actual number of cities.

In Fig. 1.5 the running time of the TSP algorithm, measured by the number of times a position of some city is assigned, when creating permutations, is shown as a function of the parameter r. One observes that, close to the point, where exactly half of the instances have a shorter (minimum tour) length than l, the running time is maximal. This means that close to the phase transition, the problem is hardest to solve. It is a well known phenomenon in physics that, for the computer simulations of phase diagrams, the running time is maximal close to phase transitions. It is one of the main purposes of this book, to understand how this coincidence arises for combinatorial optimization problems.

Finally, we mention that it is easy to understand why the running time is small for large as well as for small values of Φ (i. e., l):

- For very small values of l, even the two closest cities have a larger distance than the given value of l, hence the algorithm can stop immediately after one step.
- For very large values of l, even the first permutation of the cities has a total distance smaller than l, i. e., the algorithm stops after $n - 1$ steps.

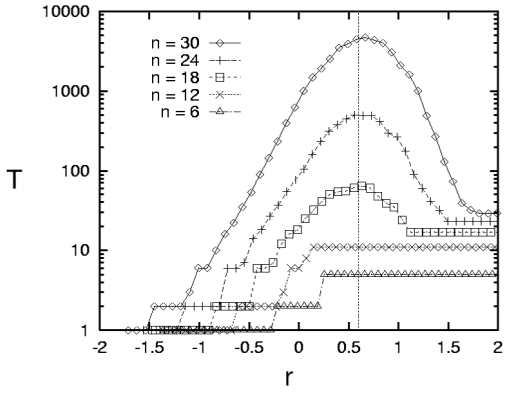

Figure 1.5: Running time for branch-and-bound algorithm as a function of $r = (l/\sqrt{nA} - 0,78)n^{2/3}$, for different number n of cities. The vertical line indicates where half of the instances have a shorter tour length than l. Reprinted from Ref. [9] with permission from Elsevier.

□

A better understanding of problems and algorithms may help to approach the solution of the ultimate task: to find fast algorithms such that larger instances can be handled efficiently. This is the main part of the work of computer scientists working in algorithmic design. Also in this field, physics has recently proved its strength. For example the *simulated annealing* technique originates in physics, see Chap. 2. The basic idea is to mimic the experimental cooling of a system to find the low-temperature, i. e., low-energy, behavior. This technique has the advantage of being widely applicable in very different fields. The disadvantage is that it does not guarantee finding the global optimum. For many problems in practice, it is, however, sufficient to find some local optima, which are close to the global one. Much more recently, the deep insight into the structural properties of the solutions of several hard optimization problems gained by statistical-mechanics analysis, has led to the proposal of a new class of statistical inference algorithms, called *survey propagation*. These algorithms are discussed in detail in Chaps 9 and 10.

1.3 Textbooks

In the following list, some useful basic textbooks are suggested.

- T. H. Cormen, S. Clifford, C. E. Leiserson, R. L. Rivest: *Introduction to Algorithms*, MIT Press, 2001
  This book is considered as the standard introduction to algorithms and data structures and provides a good foundation in the field which proves to be useful for theoretical studies and especially for practical applications and implementations of algorithms at all levels.

- A. V. Aho, J. E. Hopcroft, J. D. Ullman: *The design and analysis of computer algorithms*, Addison-Wesley, 1974
  This book merges the field of practical applications of algorithms and theoretical studies in this field.

- M. R. Garey and D. S. Johnson: *Computers and intractability*, Freeman, New York, 1979
  This book is a traditional standard text book on complexity theory. It concentrates on the part of theoretical computer science related to hard problems. The basic classes of problems are defined, many fundamental problems are explained, and their relationship is proved. In addition, the book contains a huge list of combinatorial problems which may serve as a source of inspiration for further research.

- C. H. Papadimitriou and K. Steiglitz: *Combinatorial Optimization*, Prentice-Hall, 1982
  This book gives a good introduction to the field of combinatorial optimization. All relevant basic problems and algorithms are explained. It exists in an economic paperback edition.

- A. K. Hartmann and H. Rieger: *Optimization Algorithms in Physics*, Wiley-VCH, Berlin, 2001
  This text book shows how optimization algorithms can be applied to many problems in physics. It explains the transformations needed to convert physical problems into optimization problems, and presents the algorithms needed to solve these problems.

- M. Mézard, G. Parisi, and M. A. Virasoro: *Spin glass theory and beyond*, World Scientific, Singapore, 1987.
  This book gives an introduction to the statistical-mechanics theory of spin glasses, together with reprints of the most important papers. It discusses also first applications of spin-glass methods to non-physical problems, including neural networks and combinatorial optimization.

- K. H. Fisher and J. A. Hertz: *Spin Glasses*, Cambridge University Press, 1991
  This book introduces the methods of statistical physics needed to study phase transitions in optimization problems. In this text, all techniques are applied to spin glasses, for which the methods were originally developed.

Bibliography

[1] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Science **220**, 671 (1983).

[2] M. Mézard and G. Parisi, J. Physique **47**, 1285 (1986).

[3] N. J. Cerf, J. Boutet de Monvel, O. Bohigas, O. C. Martin, and A. G. Percus, Journal de Physique I **7**, 117 (1997).

[4] A. G. Percus and O. C. Martin, J. Stat. Phys. **94**, 739 (1999).

[5] A. Chakraborti and B. K. Chakrabarti, Eur. Phys. J. B **16**, 677 (2000).

[6] D. S. Dean, D. Lancaster, and S. N. Majumdar, arXiv preprint `cond-mat/0411111`.

[7] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).

[8] K. H. Fisher and J. A. Hertz, *Spin Glasses* (Cambridge University Press, 1991).

[9] I. P. Gent and T. Walsh, Artif. Intell. **88**, 349 (1996).

2 Algorithms

This chapter gives a short introduction to algorithms. We first present a notation called *pidgin Algol* which will be used throughout this book to describe algorithms. Next, we introduce some basic principles of algorithms frequently encountered later on when studying optimization techniques: iteration, recursion, divide-and-conquer, dynamic programming and backtracking. Since there are many specialized textbooks in this field [1–3] we will demonstrate these fundamental techniques by presenting only a few simple examples.

2.1 Pidgin Algol

The algorithms will not be presented using a specific programming language. Instead, we will use a *notation* for algorithms called *pidgin Algol*, which resembles modern high-level languages like Algol, Pascal or C. But unlike any conventional programming language, variables of an arbitrary type are allowed, e. g., they can represent numbers, strings, lists, sets or graphs. It is not necessary to declare variables and there is no strict syntax.

For the definition of pidgin Algol, we assume that the reader is familiar with at least one high-level language and that terms like *variable*, *expression*, *condition* and *label* are clear. A pidgin Algol program is a sequence of *statements* between a **begin** and an **end**, or, in other words a *compound* statement (see below). It is composed of the following building blocks:

1. *Assignment*
   variable := *expression*

   A value is assigned to a variable. Examples: $a := 5 * b + c$, $A := \{a_1, \ldots, a_n\}$

   Also more complex and informal structures are allowed, like
   let z be the first element of the queue Q

2. *Condition*
   if *condition* **then** *statement 1*
   else *statement 2*

   The **else** clause is optional. If the condition is true, statement 1 is executed, otherwise statement 2 is executed (if it exists).

   Example: **if** *money*$>$100 **then** *restaurant* := 1 **else** *restaurant* := 0

3. *Cases*
   case: *condition 1*
   statement1_A;
   statement1_B;
   ...
   case: *condition 2*
   statement2_A;
   statement2_B;
   ...
   case: *condition 3*
   statement3_A;
   statement3_B;
   ...
   ...
   end cases

   This statement is useful if many different case can occur, thus making a sequence of **if** statements too complex. If condition 1 is true, then the first block of statements is executed (here no **begin** ... **end** is necessary). If condition 2 is true, then the second block of statements is executed, etc.

4. *While loop*
   while *condition* **do** *statement*

   The statement is performed as long as the condition is true.

   Example: **while** *counter* < 200 **do** *counter* := *counter*+1;

5. *For loop*
   for *list* **do** *statement*

   The statement is executed for all parameters in the list. Examples:

   for $i := 1, 2, \ldots, n$ **do** *sum* := *sum*$+i$

   for all elements q of queue Q **do** *waits*[q] := *waits*[q]+1

6. *Goto statement*
   a) *label: statement*
   b) **goto** *label*

   When the execution of an algorithm reaches a goto statement the execution is continued at the statement which carries the corresponding label.

7. *Compound statement*
   begin
   statement 1;
   statement 2;
   ...
   statement n;
   end

The compound statement is used to convert a sequence of statements into one statement. It is useful, e. g., if a for-loop should be executed for a body of several statements.

Example:
for $i := 1, 2, \ldots, n$ **do**
begin
$a := a + i$;
$b := b + i * i$;
$c := c + i * i * i$;
end

For brevity, sometimes a compound statement is written as a list of statements in one line, without the **begin** and **end** keywords.

8. *Procedures*
procedure *procedure-name* (*list of parameters*)
begin
statements
return *expression*
end

The **return** statement is optional. Note that we use the **return** statement also inside algorithms to return the final result. A procedure is used to define a new name for a collection of statements. A procedure can be invoked by writing: *procedure-name* (*arguments*)

Example:

procedure minimum (x, y)
begin
if x>y **then return** y
else return x
end

9. *Comments*
comment *text*

Comments are used to explain parts of an algorithm, i. e., to aid in its understanding. Sometimes a comment is given at the right end of a line without the **comment** keyword.

10. *Miscellaneous statements*: practically any text which is self-explanatory is allowed. Examples:

Calculate determinant D of matrix M
Calculate average waiting time for queue Q

As a first example we present a simple heuristic for the TSP problem, which we have introduced in Sec. 1.1. This method constructs a tour which is quite short, but it does not guarantee to find the optimum. The basic idea is to start at a randomly chosen city. Then iteratively the

city which has the shortest distance from the present city, i. e., its *nearest neighbor*, is chosen from the set of cities which have not yet been visited. The array v will be used to indicate which cities already belong to the tour. In σ_i the ith city visited is stored. Please remember that $d(i, j)$ denotes the distance between cities i and j and n is the number of cities.

algorithm TSP-nearest-neighbor($n, \{d(i, j)\}$)
begin
  for $i := 1, 2, \ldots, n$ **do**
    $v[i] := 0$;
  σ_1 := one arbitrarily chosen city;
  $v[\sigma_1] := 1$;
  for $i := 2, 3, \ldots, n$ **do**
  begin
    min $:= \infty$;
    for all j with $v[j] = 0$ **do**
      if $d(\sigma_{i-1}, j) <$ min **then**
        min $:= d(\sigma_{i-1}, j)$; $\sigma_i := j$;
    $v[\sigma_i] := 1$;
  end
end

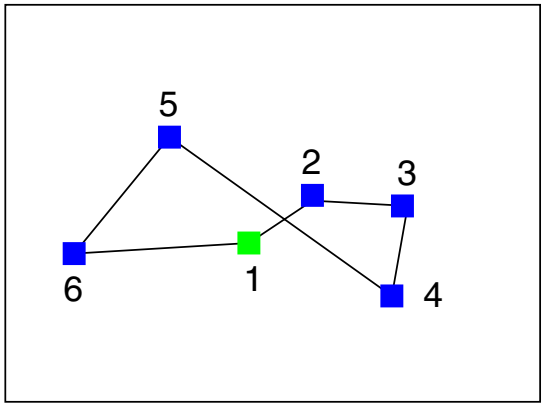

Figure 2.1: Example of where the heuristic fails to find the shortest round trip.

Please note that the length of the tour constructed in this way depends on the city where the tour starts and that this city is randomly chosen. As already mentioned, the algorithm does not guarantee to find the shortest tour. This can be seen in Fig. 2.1, where the algorithm is applied to a sample. In the resulting tour, two edges ($(1, 2)$ and $(4, 5)$) cross, which shows that a shorter round trip exists. One can replace the two crossing edges by $(1, 4)$ and $(2, 5)$.