The machine does not isolate us
from the great problems of life
but plunges us more deeply into them.
—Antoine de Saint-Exupéry, Pilot & Writer

1 Introduction

This introductory chapter provides the background and motivation for studying stochastic local search algorithms for combinatorial problems. We start with an introduction to combinatorial problems and present SAT, the satisfiability problem in propositional logic, as well as TSP, the travelling salesman problem, as the central problems used for illustrative purposes throughout the first part of this book. This is followed by a short introduction to computational complexity. Next, we discuss and compare various fundamental search paradigms, including the concepts of systematic and local search, after which we formally define and discuss the notion of stochastic local search, one of the practically most important and successful approaches for solving hard combinatorial problems.

1.1 Combinatorial Problems

Combinatorial problems arise in many areas of computer science and other disciplines in which computational methods are applied, such as artificial intelligence, operations research, bioinformatics and electronic commerce. Prominent examples are tasks such as finding shortest or cheapest round trips in graphs, finding models of propositional formulae or determining the 3D-structure of proteins. Other well-known combinatorial problems are encountered in planning, scheduling, time-tabling, resource allocation, code design, hardware design and genome sequencing. These problems typically involve finding groupings, orderings or assignments of a discrete, finite set of objects that satisfy certain conditions or constraints. Combinations of these *solution components* form the potential solutions of a combinatorial problem. A scheduling problem, for instance, can be

There is no higher or lower knowledge,
but one only, flowing out of experimentation.
—Leonardo da Vinci, Inventor & Artist

4 Empirical Analysis of SLS Algorithms

In this chapter, we discuss methods for empirically analysing the performance and behaviour of stochastic local search algorithms. Most of our general considerations and all empirical methods covered in this chapter apply to the broader class of (generalised) Las Vegas algorithms, which contains SLS algorithms as a subclass. After motivating the need for a more adequate empirical methodology and providing some general background on Las Vegas algorithms, we introduce the concept of run-time distributions (RTDs), which forms the basis of the empirical methodology presented in the following. Generally, this RTD-based analysis technique facilitates the evaluation, comparison and improvement of SLS algorithms for decision and optimisation problems; specifically, it can be used for obtaining optimal parameterisations and parallelisations.

4.1 Las Vegas Algorithms

Stochastic Local Search algorithms are typically incomplete when applied to a given instance of a combinatorial decision or optimisation problem; there is no guarantee that an (optimal) solution will eventually be found. However, in the case of a decision problem, if a solution is returned, it is guaranteed to be correct. The same holds for the decision variants of optimisation problems. Another important property of SLS algorithms is the fact that, given a problem instance, the time required for finding a solution (in case a solution is found) is a random variable. These two properties, correctness of the solution computed and run-times

Perfection has been attained
not when nothing remains to be added
but when nothing remains to be taken away.
—Antoine de Saint-Exupéry, Pilot & Writer

2 SLS Methods

Stochastic Local Search (SLS) is a widely used approach to solving hard combinatorial optimisation problems. Underlying most, if not all, specific SLS algorithms are general SLS methods that can be applied to many different problems. In this chapter we present some of the most prominent SLS methods and illustrate their application to hard combinatorial problems, using SAT and TSP as example domains.

The techniques covered here range from simple iterative improvement algorithms to complex SLS methods, such as Ant Colony Optimisation and Evolutionary Algorithms. For each of these SLS methods, we motivate and describe the basic technique and discuss important variants. Furthermore, we identify and discuss important characteristics and features of the individual methods and highlight relationships between them.

2.1 Iterative Improvement (Revisited)

In Chapter 1, Section 1.5, we introduced Iterative Improvement as one of the simplest, yet reasonably effective SLS methods. We have pointed out that one of the main limitations of Iterative Improvement is the fact that it can, and often does, get stuck in local minima of the underlying evaluation function. Here, we discuss how using larger neighbourhoods can help to alleviate this problem without rendering the exploration of local neighbourhoods prohibitively expensive.

The purpose of models is not to fit the data
but to sharpen the questions.
—Samuel Karlin, Mathematician & Bioinformatician

3 Generalised Local Search Machines

In this chapter, we introduce Generalised Local Search Machines (GLSMs), a formal framework for stochastic local search methods. The underlying idea is that most efficient SLS algorithms are obtained by combining simple (pure) search strategies using a control mechanism; in the GLSM model, the control mechanism is essentially realised by a non-deterministic finite state machine. GLSMs provide a uniform framework capable of representing most modern SLS methods in an adequate way; they facilitate representations which clearly separate between search and search control.

After defining the basic GLSM model, we establish the relation between our definition of stochastic local search algorithms and the GLSM model. Next, we discuss several aspects of the model, such as state types, transitions types and structural GLSM types; we also show how various well-known SLS methods can be represented in the GLSM framework. Finally, we address extensions of the basic GLSM model, such as co-operative, learning and evolutionary GLSMs.

3.1 The Basic GLSM Model

Many high-performance SLS algorithms are based on a combination of several simple search strategies, such as Iterative Best Improvement and Random Walk or the subsidiary local search and perturbation procedures in Iterated Local Search. Such algorithms can be seen as operating on two levels: at a lower level, the underlying simple search strategies are executed, while activation of and transitions between different strategies is controlled at a higher level. The main

[. . .] to the traveler, a mountain outline varies with every step,
and it has an infinite number of profiles,
though absolutely but one form.
Even when cleft or bored through
it is not comprehended in its entireness.
—Henry David Thoreau, Writer & Philosopher

5 Search Space Structure and SLS Performance

The performance of SLS algorithms crucially depends on structural aspects of the spaces being searched. Studying the nature of this dependency can significantly improve our understanding of SLS behaviour and facilitate the further improvement and successful application of SLS methods.

In this chapter, we introduce various aspects of search space structure and discuss their impact on SLS performance. These include fundamental properties of a given search space and neighbourhood graph, such as size, connectivity, diameter and solution density, as well as global and local properties of the search landscapes encountered by SLS algorithms, such as the number and distribution of local minima, fitness distance correlation, measures of ruggedness, and detailed information on the plateau and basin structure of the given space.

Some of these search space features can be determined analytically, but most have to be measured empirically, often involving rather complex search methods. We exemplify the type of results obtainable from such analyses of search space features and their impact on SLS performance for our standard example problems, SAT and TSP.

5.1 Fundamental Search Space Properties

The search process carried out by any SLS algorithm when applied to a given problem instance π can be seen as a walk on the neighbourhood graph associated with π, $G_N(\pi)$. Recall from Chapter 1, Section 1.5 that $G_N(\pi) := (S(\pi), N(\pi))$,

In the mountains of truth you never climb in vain:
either you reach new heights today
or you practice your strength
so you can climb higher tomorrow.
—Friedrich Nietzsche, Philosopher

6 Propositional Satisfiability and Constraint Satisfaction

The Satisfiability Problem in Propositional Logic (SAT) is a conceptually simple combinatorial decision problem that plays a prominent role in complexity theory and artificial intelligence. To date, stochastic local search methods are among the most powerful and successful methods for solving large and hard instances of SAT. In this chapter, we first give a general introduction to SAT and motivate its relevance to various areas and applications. Next, we give an overview of some of the most prominent and best-performing classes of SLS algorithms for SAT, covering algorithms of the GSAT and WalkSAT architectures as well as dynamic local search algorithms. We discuss important properties of these algorithms — such as the PAC property — and outline their empirical performance and behaviour.

Constraint Satisfaction Problems (CSPs) can be seen as a generalisation of SAT; they form an important class of combinatorial problems in artificial intelligence. In the second part of this chapter, we introduce various types of CSPs and give an overview of prominent SLS approaches to solving these problems. These approaches include encoding CSP instances into SAT and solving the encoded instances using SAT algorithms, various generalisations of SLS algorithms for SAT and native CSP algorithms.

6.1 The Satisfiability Problem

As motivated and formally defined in Chapter 1 (page 17*ff.*), the Satisfiability Problem in Propositional Logic (SAT) is to decide for a given propositional

It is the mark of an educated mind
to rest satisfied with the degree of precision
which the nature of the subject admits
and not to seek exactness
where only an approximation is possible.
—Aristotle, Philosopher

7 MAX-SAT and MAX-CSP

MAX-SAT and MAX-CSP are the optimisation variants of SAT and CSP. These problems are theoretically and practically interesting, because they are among the conceptually simplest combinatorial optimisation problems, yet instances of optimisation problems from many application domains can be represented as MAX-SAT or MAX-CSP instances in an easy and natural way. SLS algorithms are among the most powerful and successful methods for solving large and hard MAX-SAT and MAX-CSP instances.

In this chapter, we first introduce MAX-SAT. Next, we present some of the best-performing SLS algorithms for various types of MAX-SAT instances and give an overview of results on their behaviour and relative performance. In the second part of this chapter, we introduce MAX-CSP and discuss SLS methods for solving the general problem as well as the closely related overconstrained pseudo-Boolean and integer optimisation problems.

7.1 The MAX-SAT Problem

MAX-SAT can be seen as a generalisation of SAT for propositional formulae in conjunctive normal form in which, instead of satisfying all clauses of a given CNF formula F with n variables and m clauses (and hence F as a whole), the objective is to satisfy as many clauses of F as possible. A solution to an instance of this problem is a variable assignment (i.e., a mapping of variables in F to truth values), that satisfies a maximal number of clauses in F.

Traveller, there is no path,
paths are made by walking.
—Antonio Machado, Poet

8 Travelling Salesman Problems

The Travelling Salesman Problem (TSP) is probably the most widely studied combinatorial optimisation problem and has attracted a large number of researchers over the last five decades. Work on the TSP has been a driving force for the emergence and advancement of many important research areas, such as stochastic local search or integer programming, as well as for the development of complexity theory. Apart from its practical importance, the TSP has also become a standard testbed for new algorithmic ideas.

In this chapter we first give a general overview of TSP applications and benchmark instances, followed by an introduction to the most basic local search algorithms for the TSP. Based on these algorithms, several SLS algorithms have been developed that have greatly improved the ability of finding high quality solutions for large instances. We give a detailed overview of iterated local search algorithms, which are currently among the most successful SLS algorithms for large TSP instances, and present several prominent, high-performance TSP algorithms that are based on population-based SLS methods. While most of this chapter focuses on symmetric TSPs, we also discuss aspects that arise in the context of solving asymmetric TSPs.

8.1 TSP Applications and Benchmark Instances

Given an edge-weighted, completely connected, directed graph $G := (V, E, w)$, where V is the set of $n := \#V$ vertices, E the set of (directed) edges, and $w : E \mapsto \mathbb{R}^+$ a function assigning each edge $e \in E$ a weight $w(e)$, the Travelling

There is a time for some things,
and a time for all things;
a time for great things,
and a time for small things.
—Miguel de Cervantes Saavedra, Writer

9 Scheduling Problems

Scheduling is a ubiquitous task in a wide range of real-world settings and forms one of the most important classes of combinatorial problems. SLS algorithms are commonly and very successfully used for solving scheduling problems in practice. We begin this chapter with an introduction to scheduling problems and an overview of the different types of problems that fall into the scheduling domain. We then present and discuss stochastic local search algorithms for various important and prominent types of scheduling problems: single-machine, flow shop and group shop problems. As we will show, some approaches, issues and results are similar for the various types of scheduling problems, while others differ considerably. Given the variety of scheduling problems and SLS approaches for solving them, this chapter can merely provide an introduction and highlight some important issues. The interested reader will find pointers to the literature on scheduling problems in the 'Further Readings and Related Work' section at the end of this chapter.

9.1 Models and General Considerations

Scheduling problems arise in virtually all situations where performing a given set of actions or operations requires the allocation of resources and time slots subject to certain feasibility and optimisation criteria. Scheduling problems are often difficult to solve, because resources are usually scarce and complex dependencies may exist between the actions. As an example, consider the scheduling of landings and takeoffs at an airport. Here, the (typically scarce) resources are the runways.

Problems worthy of attack
prove their worth by fighting back.
—Piet Hein, Poet & Scientist

10 Other Combinatorial Problems

The problems covered in the previous chapters are only some of many combinatorial problems to which stochastic local search algorithms have been applied successfully. In this chapter, we present and discuss SLS applications to other combinatorial problems, which have been selected partly because of their fundamental nature, partly because of their relevance for certain application areas. In each of the main sections, we will introduce one combinatorial problem, discuss its applications and commonly used benchmark instances, and present one or more SLS approaches for solving this problem. The problems we cover are: Graph Colouring, Quadratic Assignment, Set Covering, Combinatorial Auctions Winner Determination and DNA Code Design. While the first three problems have been extensively studied in the literature for many years, the latter two have only relatively recently gained their current prominence.

The algorithms presented in this chapter are primarily intended to illustrate the application of SLS methods to the respective problems. Especially for the Graph Colouring Problem, the Quadratic Assignment Problem and the Set Covering Problem, the algorithms we present were chosen from a large number of known SLS algorithms for the respective problem; our selection represents a compromise between our desire to present state-of-the-art algorithms and the need to give reasonably didactic examples. References to other SLS algorithms and more detailed information on the problems covered here are provided in the 'Further Readings and Related Work' section of this chapter.

We shall not cease from exploration
and the end of all our exploring
will be to arrive where we started
and to know the place for the first time.
—T.S. Eliot, Poet

EPILOGUE

Let us return to Augsburg and the problem of visiting all of its 127 Biergärten in a single day as well as to the logic puzzle, solving which would pay for all the beer along the way. It is easy to see that the former problem can be formalised as a TSP instance (see Chapters 1 and 8), while the latter can be modelled as an instance of SAT or CSP (see Chapters 1 and 6). Consequently, they could be solved using a variety of SLS algorithms for these problems (see Chapters 2, 6 and 8). Given the relatively small instance sizes, even relatively simple SLS algorithms would solve these problems easily, at least when run on a modern computer. (Clearly, many other search methods could be used to solve these problems similarly efficiently.)

Generally, Moore's Law, that is, the exponential increase over time in processing power and memory size (or more precisely: memory density) that occurred since the beginning of modern computing, plays a considerable role in pushing the limits of practical solvability of hard combinatorial problems. More importantly, at least in the case of SLS methods, it facilitates the rapid evaluation of algorithmic behaviour and makes possible new forms of empirical analysis, which provide the basis for the development of improved algorithms. Both factors combined, hardware and algorithmic improvements, have led to dramatic progress in our ability to solve large instances of hard combinatorial problems; and in many cases, SLS methods provide the algorithmic basis for these developments.

Nevertheless, many challenges remain in solving hard combinatorial problems in practice. There are problems, such as the QAP (see Chapter 10, Section 10.2), for which even relatively small instances are beyond the reach of state-of-the-art SLS methods (and any other algorithmic technique). Tight time